An Investigation into Dynamic Web Service Composition Using a Simulation Framework

YOUSIF-MOHAMMAD, KHALID,MIRGHNEE

An Investigation into Dynamic Web Service Composition Using a Simulation Framework

Khalid Mirghnee Yousif Mohammad

A Thesis presented for the degree of

Doctor of Philosophy

School of Engineering & Computing Sciences

Durham University

November 2012

Abstract

[**Motivation**] Web Services technology has emerged as a promising solution for creating distributed systems with the potential to overcome the limitation of former distributed system technologies. Web services provide a platform-independent framework that enables companies to run their business services over the internet. Therefore, many techniques and tools are being developed to create business to business/business to customer applications. In particular, researchers are exploring ways to build new services from existing services by dynamically composing services from a range of resources. [**Aim**] This thesis aims to identify the technologies and strategies currently being explored for organising the dynamic composition of Web services, and to determine how extensively each of these has been demonstrated and assessed. In addition, the thesis will study the matchmaking and selection processes which are essential processes for Web service composition. [**Research Method**] We undertook a mapping study of empirical papers that had been published over the period 2000 to 2009. The aim of the mapping study was to identify the technologies and strategies currently being explored for organising the composition of Web services, and to determine how extensively each of these has been demonstrated and assessed. We then built a simulation framework to carry out some experiments on composition strategies. The first experiment compared the results of a close replication of an existing study with the original results in order to evaluate our close replication study. The simulation framework was then used to investigate the use of a QoS model for supporting the selection process, comparing this with the ranking technique in terms of their performance. [**Results**] The mapping study found 1172 papers that matched our search terms, from which 94 were classified as providing practical demonstration of ideas related to dynamic composition. We have analysed 68 of these in more detail. Only 29 provided a 'formal' empirical evaluation. From these, we selected a 'baseline' study to test our simulation model. Running the experiments using simulated datasets have shown that in the first experiment the results of the close replication study and the original study were similar in terms of their profile. In the second experiment, the results demonstrated that the QoS model was better than the ranking mechanism in terms of selecting a composite plan that has highest quality score. [**Conclusions**] No one approach to service composition seemed to meet all needs, but a number has been investigated more. The similarity between the results of the close replication and the original study showed the validity of our simulation framework and a proof that the results of the original study can be replicated. Using the simulation it was demonstrated that the performance of the QoS model was better than the ranking mechanism in terms of the overall quality for a selected plan. The overall objectives of this research are to develop a generic life-cycle model for Web service composition from a mapping study of the literature. This was then used to run simulations to replicate studies on matchmaking and compare selection methods.

Acknowledgements

I would like to express my thanks to:

- A Supervisor and role model, Prof. David Budgen, for his dedicated supervision, critical questioning, continuing support and for investing plenty of time and effort to make my project a success.
- To my beloved family: my father, Mirghnee, my mother Qamer, my brothers Yousif and Mohammad my sister Moahib, for their support and love.

This work was funded partially by a postgraduate scholarship from the Ministry of Higher Education, Republic of Sudan.

Declaration

I hereby declare that this thesis is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

This chapter introduces the work that is described in this thesis. The research goal has been to investigate the Dynamic Web Service Composition (DWSC) process. Main contributions of this thesis is consisted of conducting a Systematic Literature Review (SLR) to identify what has been implemented, together with performing experiments by using a simulation framework to study the matchmaking and selection processes that are involved in DWSC. This chapter also describes the structure of the thesis and provides a short overview of the topics covered in each chapter.

1.2 Motivation

The emergence of Web services as a platform-independent basis for realising distributed systems provides companies with the opportunity to run their business by using and providing services over the Internet (Medjahed et al., 2003). The emergence of XML and of XML-based languages, providing platform-independence over many Operating Systems, has had great impact on the success of Web services in particular by using such forms as SOAP-based service over HTTP to build distributed systems (Dustdar and Schreiner, 2005). In general, a Web Service is a self-contained component that can be provided by

heterogeneous and multiple providers, with its ultimate design goal being to improve interoperability among applications within organisations or across organisations (Gailey, 2004). If comparing Web Services with other technologies for developing distributed components, such as DCOM and COBRA, Web Services offer lower cost and faster software development in terms of reuse (Potts and Kopack, 2003). Web services can co-operate over the Internet, regardless of the tools being used to develop each of them. Therefore, there may be no need to develop new systems from scratch (Potts and Kopack, 2003). This feature has made it possible to integrate heterogeneous distributed systems without taking into account their compatibility isssue. Many techniques and tools are being developed for building those systems or connecting business-to-business and business-to-customer applications using distributed service models (Hamadi and Benatallah, 2003). As an important element of this, researchers also explore ways to build new services from existing services by composing services from a range of resources. Figure 1.1 gives an idea of mapping a business plan into a service composition plan.

The emergence of standards and languages that support Web service composition allow many promising approaches to be explored (Milanovic and Malek, 2004). REST is another technology that has recently emerged for developing Web Services (Fielding and Taylor, 2000). REST relies on a set of architectural principles by which developers can create Web services that concentrate on models of a system that involve how resource states are recognised and transferred[1] over HTTP by a number of various clients in different platforms (Fielding and Taylor, 2000, Rodriguez, 2008). This thesis focuses mainly on SOAP-based Web services since these are more established forms that are widely in the literature.

The way services are linked together can be classified as either *static composition*, which involves binding requests to specific services at construction time, or *dynamic composition*, where they are bound at run time (i.e. when the service is delivered) (Dustdar and Schreiner, 2005). Static composition has many parallels with component-based development, where the processes of discovery, selection and composition are performed by the developer in advance of ac-

[1] A representation used to capture the current or intended state of that resource and transferring that representation between components.

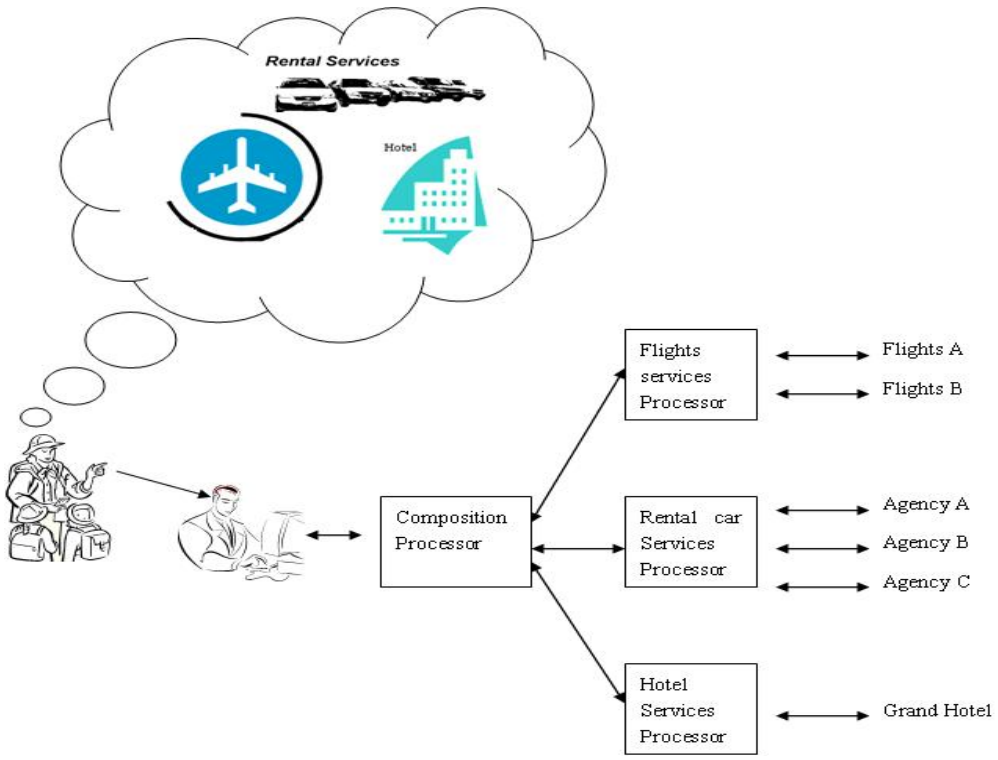


Figure 1.1: Converting business ideas to real service composition plan

tual use (Sun, 2004, Nie et al., 2006). However, while for static composition the use of services as design elements offers a whole new set of challenges in terms of design criteria, notations and architectural models. These are essentially challenges that need to be met through the use of human skills. Dynamic composition then poses an extended set of challenges that requires the relevant decisions to be made with a minimum of human intervention. These decisions will need to be made to at least an adequate standard if a system is to function. Engaging Web services in a composition plan requires agreement/contract between a service provider and a service requester which involves negotiation between these two parties. Negotiation permits the provider to give to the requester an entry-point for getting information about a Web service (Kerrigan, 2006, Skogsrud et al., 2004). The service provider gives permission to others in order to access/use the service and applies a charge for use. The service requester agrees about the functionality and quality of the service to be delivered by the provider, and pays the bills for using the service.

Both static and dynamic composition therefore involves important, but different, issues that need to be addressed by the research community. Dynamic composition particularly requires the development of suitable forms and mechanisms for describing services as well as for selecting them 'on the fly' in order to meet specific run-time requirements - both functional and non-functional.

Devising these forms and mechanisms, establishes the basis of a substantial body of literature that has been proposed by the research community . However, the observed preference of software engineering research for 'concept analysis' and 'concept implementation' as research methods (Glass et al., 2004) means that relatively little is known about such issues as how well these work, how extensively they are applicable and how they are compared in terms of non-functional requirements. Our motivation for undertaking this study was to identify the approaches to dynamic service composition that are currently being implemented. in addition to this, the study aimed to identify the extent of empirical knowledge about the most important issues that needed to be addressed in managing the process of service composition.

The research questions adopted were:

- What are the main issues that need to be addressed if dynamic service composition is to be successfully implemented and widely adopted?
- What solutions have been proposed to deal with the issues raised?
- Which research methods have been used to investigate the proposed strategies?
- What gaps are there in current research?

Our longer-term research focus is directed to investigate the benefits of using *Simulation* in studying service processes in general, and service composition in particular. A software process simulation is an abstract model which imitates all or part of a system in a simplified way, a form achievable on a computer. A system could be a real or proposed software development or an evolution process, with the simulation generating results demonstrating actual situations or the predicted results for proposed process alteration (Wernick and Hall,

2007). These results are usually exhibited in quantified terms. In general, simulation models are based on textual or graphical structures designed to represent components of the real or suggested process and of the interactions between them.

1.3 The approach (Evaluation of the Research Question)

To address the research questions, we began with a systematic literature review (SLR) in the form of a mapping study to collect empirically-based knowledge that was related to service composition (Kitchenham et al., 2011). We have chosen to use an SLR as it was rigorous and comprehensive technique for conducting a survey to identify, evaluate and interpret all available research relevant to dynamic Web service composition. The guidelines proposed by Kitchenham *et al.* (2007) stated that there are many reasons for performing an SLR, and these guidelines helped direct our study. The following are some reasons for using an SLR, as listed in the guidelines:

- To summarise the existing evidence concerning a treatment or technology, e.g. to summarise the empirical evidence of the benefits and limitations of a specific composition strategy.
- To identify any gaps in current research in order to suggest areas for further investigation.
- To provide a framework/background in order to appropriately position new research activities.

In this thesis, we have used an SLR to examine the extent to which empirical evidence is available for DWSC. Since "Web service" is widely used as a synonym for "software services", and Web Services are a widely implemented form of service, therefore, we restricted our study to this class of service. In Chapter 3 and 4 of thesis we describe how this was organised and performed. Additionally we report on the results and provide a classification of them.

From the SLR outcomes, we have found four papers that proposed life-cycle models for service composition. In these models different activities (supportive operations) are used to develop a life-cycle model for Web service composition. Through studying and analyse the activities of these models, we derived a generic life-cycle model for web service composition. This model is composed of the essential processes that can be adopted by researchers and practitioners to develop their own solutions for service composition. The model has been validated by comparing it with those four life-cycle models. Chapter 5 describes our generic life-cycle model in details.

In Chapter 7 and 8 of the thesis described an investigation of two essential processes of the service composition, matchmaking and selection processes. We performed a close replication study of using Web semantic for matchmaking process. Further, we conducted another study using QoS model for selection process in order to be integrated with the previous study. Our investigation involved number of experiments as a validation of these studies. We chose simulation for this purpose as it was one of the most widely used operations research and management science techniques (Pickard et al., 2001, Law, 2008). Banks *et al.* (2004) have identified a number of purposes for using simulation. The following list identifies some of these purposes:

- Simulation can be used to experiment with new designs or policies before implementation, so as to prepare for what might happen.
- Simulation helps with generating simulated datasets when there is no enough data to perform an experiment, or it is too expensive to work with the real data, or when new characteristics are introduced which are not in the existing data, etc.
- The knowledge gained during the designing of a simulation model could be of great value towards suggesting improvements in the system under investigation.
- Changing simulation inputs and observing the resulting outputs can produce valuable insight into which variables are the most important and into how variables interact.

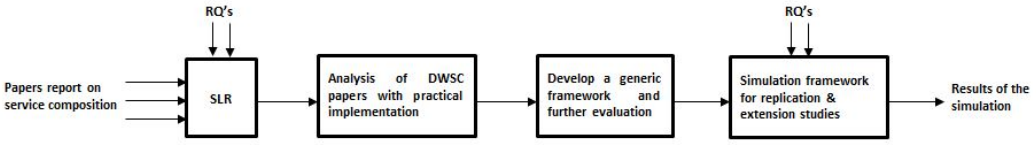


Figure 1.2: Diagram describe our research processes

- Animation shows a system in simulated operation so that the plan can be visualised.

1.4 The contribution

The contribution of this thesis takes two forms. First, it describes the conduct of a systematic literature review to investigate the available material related to Dynamic Web Service Composition. Second, it describes the creation of a simulation framework, and its use for conducting experiments to investigate same issues related to the process of matchmaking and selection of Web services to compose a new Web service (see figure 1.2).

For the systematic literature review, there are various guidelines that can be followed in order to apply the SLR in the software engineering field (Brereton et al., 1999, 2006, 2007). These papers formed the base that we used to plan our review that target the dynamic Web service composition. Our plan involved a number of sequential steps/actions that we set up to conduct our review (see Chapter 3 for more details). The outcomes of our review (SLR) explored the forms of research that was performed in the field. Furthermore, it identified the major techniques that were used to develop dynamic Web service composition and the extent to which they were evaluated and done.

Another goal for SLR was to obtain initial pointers to the most promising approaches and solutions from those that have been proposed as the basic for service composition. Through our SLR, we have also identified a number of papers that dealt with simulation and Web services from different aspects, such as adapting tools for simulation, development of service process, evaluating

approaches and simulation platform. In addition to simulation, studying the literature on the dynamic Web service composition (DWSC), has assisted with developing some ideas that were then used to help build our simulation framework. These triggered our motivation to look at how simulation can be used to study Web service composition issues. Therefore, we used the simulation framework to study and investigate the process of Web service composition, and particularly the process of matchmaking and selection of Web Services. We were interested to study the impact of using Web semantic to enhance the description of Web services and the impact of different functional and non-functional properties for these two processes. We then used simulation as the basis for conducting a number of experiments to study these factors and parameters in more detail.

In our study we required to use a datasets to run our experiment. However, the literature review showed a lack of usage of any real data-sets with the characteristics that we needed for our research. Therefore, we generated simulated data-sets with the characteristics needed for our experiments. The data-sets consisted of 30,000 service interfaces, where these interfaces used in ten service compositions scenarios in order to generate number of composition plans that intended to achieve specific different tasks. The experimental model was built around the use of a MySQL database that used to provide a registry in order to accommodate the data-sets. Java NetBeans as the platform to develop (simulate) the composition process, focused mainly on the processes of matchmaking and selection of Web services. We conducted number of experiments to study these two processes. The first part of our experiments studied the use of Web semantic for matchmaking process through a close replication study. The second part studied the use of extensible QoS model for selection process. In this study, we examined the use of five quality criteria as non-functional properties to select an optimal plan for a composite service that can achieve a specific task. We expected the results to show the impact of these parameters on the quality of the composition plan which we believed would assist developers to adopt a solution that best suited their requirements.

1.5 Thesis outline

Chapter 2 introduces related work on the topic of Web Services and their environment, including the definition of a Web Service (WS) and the related standards that have been the main factors in its success. The same chapter also defines the concept of Web Service Composition and its meaning.

Chapter 3 describes the conduct of the systematic literature review (SLR) and discusses the protocol that we designed to conduct our review.

Chapter 4 presents the results of conducting the SLR to investigate the literature that studied DWSC. It is worth mentiong that it is one of the main contributions of this thesis, as no previous study appears to have addressed this issue. The outcomes derived from this chapter, outline the key strategies used for DWSC.

Chapter 5 presents a generic framework that models the process of composition for software services. In addition, this chapter presents further analysis and summary of the empirical papers that are outlined in the previous chapter.

Chapter 6 investigates the uses of simulation as a research method for software engineering. This chapter also gives a background about replication study and its feasibility for creating and extending scientific knowledge.

Chapter 7 describes the evaluation process of a number of the identified empirical papers in order to choose a study for a replication study. Furthermore , the same chapter presents a close replication study of the chosen paper which is based on using Web semantic for matchmaking process. The outcomes of the matchmaking process is a number of composition plans which are able to achieve a same goal.

Chapter 8 describes a study of using Web semantic and QoS model to enhance the selection process. The goal of the QoS model is to offer an evaluation mechanism to differentiate between the generated composition plans. This model is based on the non-functional properties of the generated plans. Then, the generated plans are ranked according to service consumers preference in order to select an optimal plans.

Finally, in chapter 9 we discuss the work that has been done, consider threats to validity of our work and interpret the outcomes. Chapter 10 concludes the

thesis by summarising the work that has been done, in addition to identify the possibilities for future work in this area.

Chapter 2

Web Service Composition (background)

2.1 Introduction

This chapter begins with the introduction of the concepts of Web Service. Then it specifies the definition and the basic Standards for Web Services Architecture, such as XML and XML-base technologies. The chapter proceeds by describing the Web Service Environment and the characteristic of Web Service, such as autonomy, heterogeneity and dynamism. It also underlines the challenges that face the Web service development. In addition to this, the chapter also introduces the Representational State Transfer (REST) Architecture. Furthermore, the chapter brings in the concept ot Service Oriented Architecture (SOA) and its components that become a base-ground to develop distributed application over the Internet. The chapter ends with the definition and elaboration of the Web Service Composition.

2.2 Understanding Web Services

In order to understand the Web Services, it is useful to start by separating the two words, 'Web' and 'Service'. The majority of people know what the 'Web' means. While descussing the Web, the concepts such as a personal Web page, a

company Web site, E-mail and Web sites for entertainment immediately come to mind. In certain ways people they also know what a service is, and maybe thought of this in terms of the way that businesses provide services to other businesses or customers. Therefore, the growth of the Web and its applications can assist and develop businesses by running their services over the internet. Thus, by bringing these two concepts together, people have started to use the idiom 'Web Service' that can be used to describe an emerging concept that can be used through the development of the Internet for running distributed applications across organisations (that have a variety of systems) for implementing business collaborations (Potts and Kopack, 2003). To understand the concept of a Web Service well, this thesis defines the Web Service in various ways, dealing with the things that have contributed to, and developed from, the concept of the Web Service.

2.2.1 The Definition and the basic Standards for Web Services Architecture

Relatively speaking, a Web Service (WS) is still a new technology in the software engineering discipline. A WS is a self-contained piece of software which can be viewed as a procedure or subprogram that can be accessed over the Internet via common communication protocols, which make the accessibility to the WS possible through its URL (Dustdar and Schreiner, 2005). Any application is considered a WS if it is accessible over the Internet via a common communication protocol, such as HTTP and using XML format for message exchange. In addition, the World Wide Web Consortium (W3C) approved many standards which played an essential role in WS development (Abrougui et al., 2009). These standards propose ways to overcome the drawbacks of older technologies for distributed systems such as handling heterogeneity. In addition, these standards make the WS into a promising solution for developing distributed systems. Moreover, by complying with these standards, developers can use their own experiences and Web technologies to develop distributed systems without concerning the interaction with heterogeneous systems. The following are the major technologies that are available for developing the Web Services:

- Xtensible Markup Language (XML) : The idea behind the design of XML is to provide generality, simplicity, and usability over the Internet (Ray, 2003). It is a textual description that is supported by Unicode to present data in a format that can be understood worldwide. While the essential aim of XML is to structure the documents, it can also be used to represent the user's defined data structure that can be used somewhere else, for example in Web services.

- Simple Object Access Protocol (SOAP) is a protocol specified for exchanging structured information needed to implement the Web Services in computer networks (W3C, 2007). It relies on XML for encoding of its message format, and usually depends on other Application Layer protocols, most notably Remote Procedure Call (RPC) and Hypertext Transfer Protocol (HTTP), for message negotiation and transmission.

- Web Service description language (WSDL) is an XML-based format for describing network services so as a set the endpoints operating on messages that contain either document-oriented or procedure-oriented information (W3C, 2001). The operations and messages are described in the abstract, which are bound to a concrete network protocol and message format that help to define the endpoint (see Figure 2.1).

- Universal Description, Discovery and Integration (UDDI) specifications define a registry service for Web services and other electronic and non-electronic services (OASIS, 2010). A UDDI registry service is a Web service that manages information about service providers, service implementations and service metadata. Service providers can use UDDI to advertise the services they offer. Service consumers can use UDDI to discover services that suit their requirements as well as to obtain the service metadata needed to consume those services.

The following Figure 2.2 gives an overview of how two applications can interact in Web service environment. Their integration layers offer a WSDL document which provides the service interface to each other for interaction.

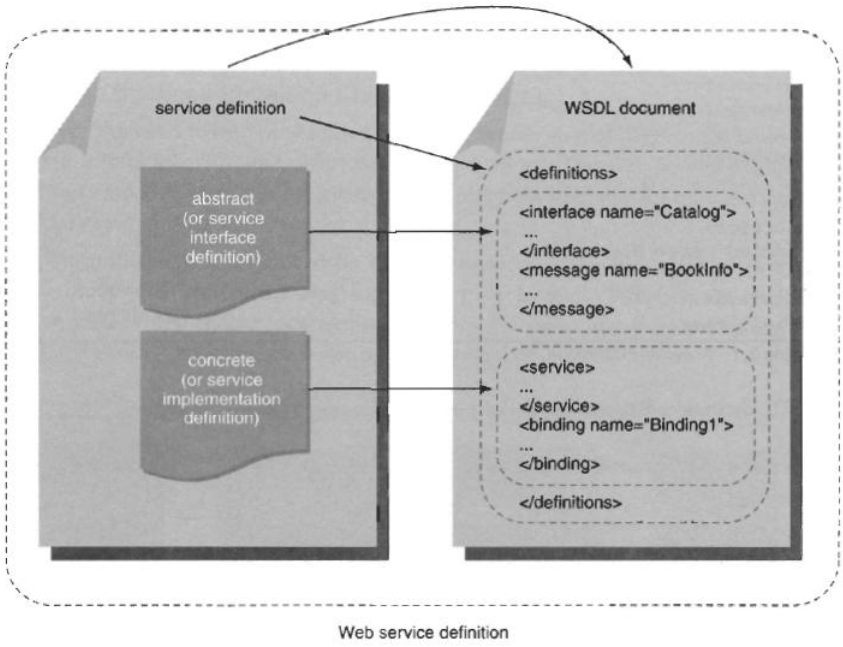


Figure 2.1: The content of a WSDL document, as the relate to a service definition (Erl, 2004).

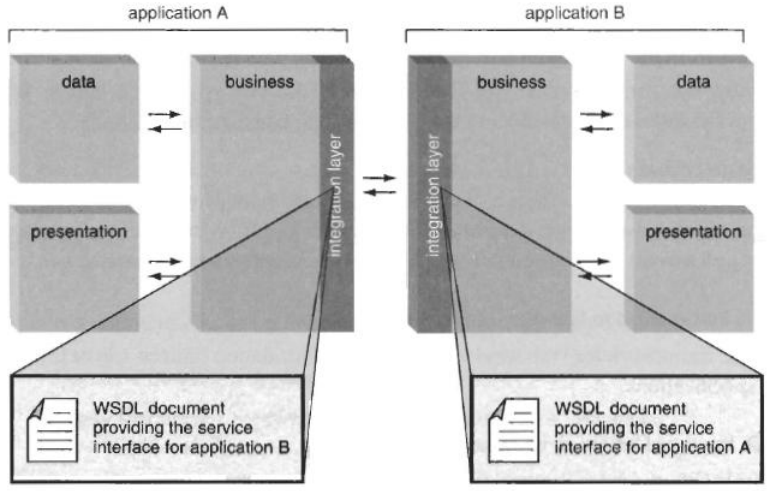


Figure 2.2: WSDL documents representing Web services to applications (Erl, 2004).

2.2.2 Web Services Environment

Web service based on open environments presents a number of challenges for developers in dealing with the autonomous and heterogeneous components. The following paragraphs discuss the concepts that are related to open information environments including autonomy, heterogeneity and dynamism of the Web Services. In order to understand and distinguish these concepts in a easy way, require them to be linked with the independence of users, designers and administrators, respectively (Singh and Huhns, 2005).

Autonomy means the components are independent from their environment which they function under their own control. Commonly, "software components are autonomous because they reflect the autonomy of the human and corporate interests that they represent on the Web. In other words, there are socio-political reasons for autonomy. Resources are owned and controlled by autonomous entities and that is why they behave autonomously" (Singh and Huhns, 2005).

Heterogeneity refers to a given system consisting of components that differ in their design and construction. Similar to the autonomy, there are two technical and socio-political reasons for heterogeneity. Component developers have complete choice to build their components in different ways in order to achieve various performance requirements. Singh and Huhns (2005) mentions that "Often, the reasons are historical: components fielded today may have arisen out of legacy systems that were initially constructed for different narrow uses, but eventually expanded in their scope to participate in the same system". Moreover, at various levels within a system heterogeneity such as data formats, encoding of information and networking protocols can be arised. Obviously, heterogeneity can be reduced by standardisation at each level. Furthermore, the standardisation can advance the productivity through enhanced interoperability. Standards always evolve and various software components may slow down behind or exceed the standards in various respects. Generally, it is easier less complicated to establish and conform to the standards of lower level.

Dynamism can be exhibited in an open environment through two main re-

spects. Firstly, due to the autonomy, components in open environment can behave arbitrarily. Especially their behaviour is subjected to change as a consequence of how they happen to be configured. Second, they may also leave or join an open environment without any explainable reasons. Singh and Huhns claim that "It is worth separating out this aspect as a reflection of the independence of the system administrators. A large-scale open system would of necessity be designed so as to accommodate the arrival, departure, temporary absence, modification, and substitution of its components" (Singh and Huhns, 2005). With respect to the first type of dynamism, each component can change dynamically in its architecture and implementation and interactions, which therefore lead to change its behaviour. That is, there may be changes in their externally observed behaviour, how they perform or deliver their behaviour, and how they collaborate with other components.

Challenges (specifically technical challenges): As WS operates in an open environment, this creates major technical challenges. In particular, it needs to accommodate the heterogeneity of the diverse participants, their independency and coordination of their actions. Because of this wide range, it is difficult to maintain awareness of all the non-functional characteristics of the available resources, such as their reliability, functionality, trustworthiness and so on. This poses significant challenges to the developers when they need to create interaction between some required resources in terms of composing a co-operative activity in order to achieve a specific task. Creating this co-operation needs the developers to know how to use them, bind them and check their compliance (Singh and Huhns, 2005).

The figure 2.3 describes how software implementations of Web services can be developed by using any programming language, operating system, or middleware system (Newcomer, 2002).

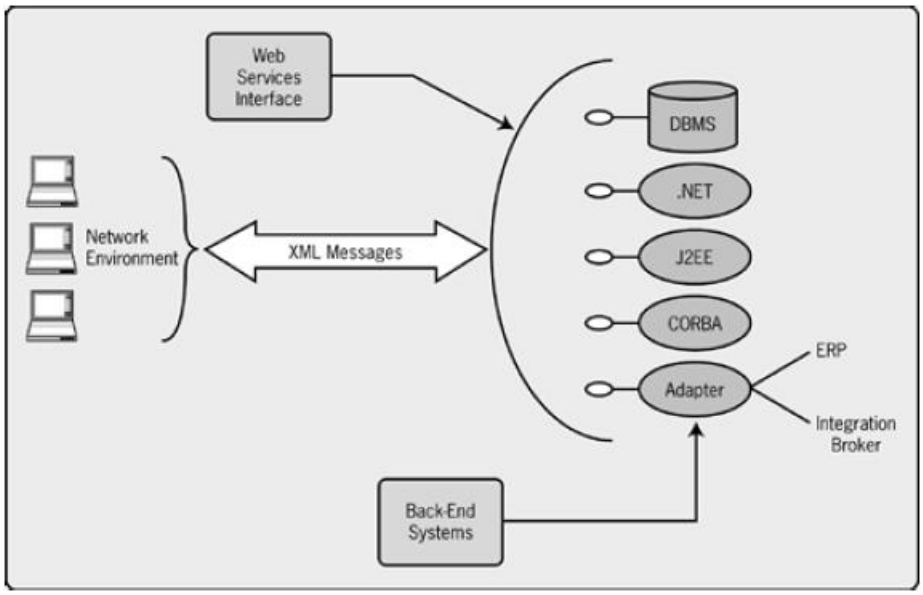


Figure 2.3: Web services interface with back-end systems (Newcomer, 2002).

2.3 Service-Oriented Architecture (SOA)

Service-Oriented Architecture (SOA) provides the conceptual framework for Web Services. Simply, a SOA consists of three entities and three operations. The entities are: service provider, service client and service discovery agent (service registry) (Potts and Kopack, 2003, Erl, 2004). Figure 2.4 describes the relation between these entities and the operations among them.

- The *Service Provider* is the element that owns the service and uses the *publish()* process to advertise its services to others via the service registry. The advertisement describes the service's capabilities and document that can be used through a contractual interface.

- The *Service Discovery Agent* is a public access catalogue which looks like a business catalogue containing information about the service providers (businesses) and descriptions of the service(s), which they can provide. This information provides contact details for these providers as well as a description of their services. Furthermore, it also contains mechanisms to organise these services and make them available to the public for inquiry.

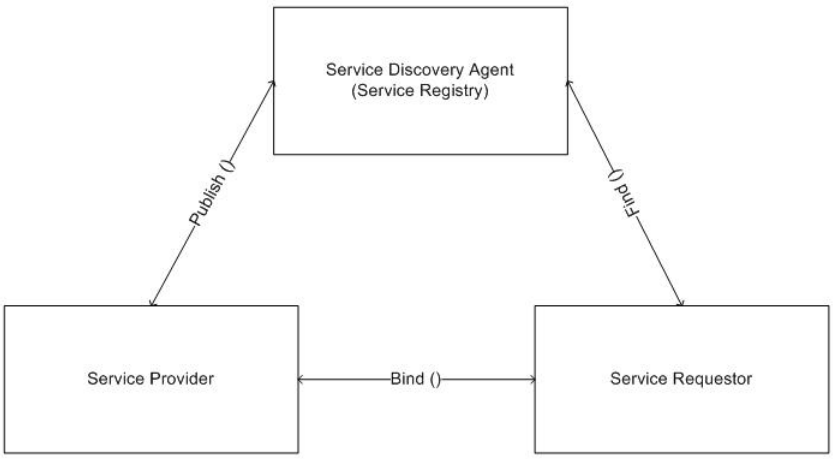


Figure 2.4: Service-Oriented Architecture: Elements and Operations

- The *Service Requester* is a business client that requires a service which can be performed by another business(s). The requester begins with the criteria for the required service and uses the `find()` process to send a query to the registry in order to discover the providers such a service. When the service requester gets the details (description/connection) about a suitable service from the registry, it uses the `bind()` process to interact with service provider. The interaction between the requester and provider can be in the form of request/response.

SOA is similar to a component architecture in terms of using similar concepts such as building blocks (service vs. component) (Stojanovic and Dahanayake, 2005). This service is reusable, same as a component, which means the same service can be reused to construct other applications. SOA forms are commonly based on XML Web service technology. So the WS plays a vtial role of a middle ground where software vendors can meet to provide openly intractable solutions which help building distributed systems.

2.4 Representational State Transfer (REST) Architecture

REST is another technology to develop Web Services that different from SOAP and WSDL based Web Services. The latter are operation centric which are developed in a Remote Procedure Call (RPC) on top of HTTP (Rauf et al., 2010, Al-Zoubi and Wainer, 2009). In-contrast, REST Web services are data centric which offer style of an architecture whose assumptions are easy to understand and design. REST advertises its functionality in the exposed resources. These resources are very similar to the object in a class that allow information on data which can be treated to answer a purpose. It has been introduced in first place in 2000 by Roy Fielding in his doctoral dissertation (Fielding and Taylor, 2000, Rodriguez, 2008). Fielding states that "(REST) architectural style, developed as an abstract model of the Web architecture and used to guide our redesign and definition of the Hypertext Transfer Protocol and Uniform Resource Identifiers. We describe the software engineering principles guiding REST and the interaction constraints chosen to retain those principles, contrasting them to the constraints of other architectural styles" (Fielding and Taylor, 2000).

REST Web services have a distinctive architectural style with different demand design conception and techniques. Conforming to the REST architectural style is generally referred to as being "RESTful". The design of a RESTful Web service interface means to offer addressability, connectivity, uniform interface and statelessness. Therefore, a composite RESTful Web service is built in a way that it applies these basic principles of a REST Web service (Al-Zoubi and Wainer, 2009). Pautasso et al. (2008) have listed four technology principles that REST architecture is based on:

- *Resource identification through URI .* A RESTful Web service reveals a group of resources which determine the goals of interaction with its clients. These resources identified with URIs, which offer wide range of addressing to identify resource and service discovery.

- *Uniform interface.* It makes use of number of request operations that are presented in HTTP, such as, `PUT`, `GET`, `POST` and `DELETE`. A new

resource that is created by `PUT` operation, can eventually be removed by `DELETE` operation. `POST` is used to update/transfer a resource into a new state. `GET` is applied to retrieve the existing state of a resource in a certain representations.

- *Self-descriptive messages.* Resources and their representation are loosely coupled, which means their contents can be accessed through a variety of formats (e.g., plain text, PDF, JPEG, HTML, XML, etc.).
- *Stateful interactions through hyperlinks.* Every interaction with a resource should have all of the required information to be processed (i.e., request messages are self-contained).

2.5 Web Service Composition (WSC)

Web Service Composition (WSC) is the process of combining different WS's together to achieve a new complex task (Bart et al., 2003, Song et al., 2006). Composition of Web service is applicable when a client's request cannot be achieved or fulfilled from the available Web services. The simplest form uses two WS's: the first WS requests a service from the second one, and the latter responds to the former's request. The service requester is responsible for formulating the composition (invocation and data flow) and is itself considered as a WS. The construction of WSC is defined on an abstract level which is based on WS's description. The actual interaction between the WS requester and the WS provider(s) takes place on-the-fly during or prior to the execution time.

There are two terms used to describe the interaction between services during the composition:Web Services *Orchestration* and *Choreography* (Peltz, 2003). The meaning of these two terms are slightly overlapping but with different messages. While the orchestration is the coordination of the Web services by the means of executable business process, the choreography is the coordination of Web services by the means of abstract business process (Peltz, 2003, Gortmaker et al., 2004). Orchestration addresses the situation of the business flow consisting of many parties. The interaction of these parties needs to be controlled by one party (defined in one place) which is the business process formulator (WS

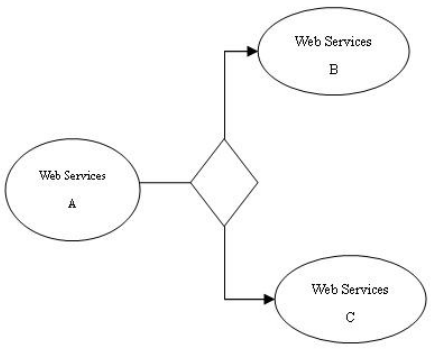


Figure 2.5: Web Service Orchestration

composer). Peltz (2003), Meng and Arbab (2007) have mentioned the two languages BPEL and BPML as tools that are used for orchestration. These tools assist the Web service composer to describe the behaviour of a business process that is composed of Web services based on the interaction, which is achieved through messages exchange and execution order (see Fig. 2.5). Choreography differs from Orchestration in terms of each WS, which involves in the composition that can interact with others WS's via message exchanges. In other words, choreography exposes the message sequences in a group of multiple parties and sources,particularly the public message exchanges which take place among Web services.Instead of a specific business process where only one party executes. WSCI and WSCDL are recommended by W3C (2002, 2004) as languages that can be used to describe peer-to-peer collaborations of parties, which in turn define the overall choreography or message exchanges (see Fig. 2.6).

2.6 Summary

In this chapter, we gave an overview of Web service and its related technologies. We started with the understanding of Web service and definition of its related technologies. In addition to this, we discussed the environment of Web service and showed that how it challenge the developers to build a new composite service system from existing heterogeneous services. , We also addressed the SOA conceptual framework as base to build such systems which we would adopt in

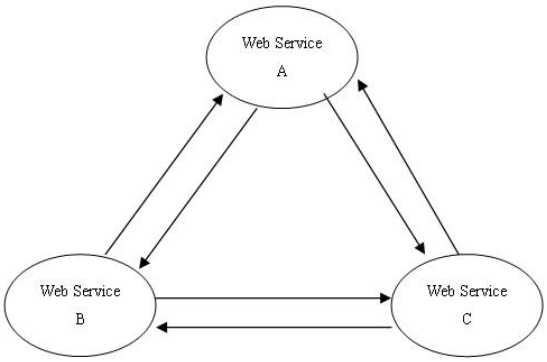


Figure 2.6: Web Service Choreography

the coming chapters. This chapter gave a brief overview of RESTful which is another paradigm to build composite service. The chapter concluded by discussing the Web service composition and its two different methods that can be used to describe the flow control of services interaction within the composition.

Chapter 3

Systematic Literature Review

3.1 Introduction

The emergence of Web services as a platform-independent basis for distributed systems encourages companies to run their business services over the Internet. Many techniques and tools are being developed for building these systems to connect business-to-business and business-to-customer applications, using distributed service models. As an important element of this, researchers are engaged to explore ways to build new service from existing services by composing services from a range of resources. In addition to this, the emergence of standards and languages that support Web service composition, allowing many promising approaches to be explored (Milanovic and Malek, 2004).

We can classify the way these services are linked together as either *static composition*, which involves binding requests to specific services at construction time, or *dynamic composition*, where they are bound at run time (i.e. when the service is delivered). Static composition has many parallels with component-based development, where the processes of discovery, selection and composition are performed by the developer in advance of actual use. This is important because the design decisions are made by a human and static composition that create few new challenges for the researchers. In contrast, for dynamic composition to be effective, some or even all of these tasks need to be performed with minimum human intervention.

This research challenge then creates the opportunity to find the best way of performing the dynamic composition of services, which may well be provided by distributed autonomous agencies and the process of composition. This involves the selection from them 'on the fly', in order to meet specific run-time requirements.

Our motivation for undertaking this review was to identify the approaches to service composition that are currently being proposed. In addition to this, we also aimed to identify the most important issues that needed to be addressed in managing the process of service composition. Another goal of this review was to obtain initial pointers to the most promising approaches and solutions from those that have been proposed as the basic for service composition. The research questions adopted were as follows:

- What are the main issues that need to be addressed if dynamic service composition is to be successfully implemented and widely adopted?
- What solutions have been proposed to deal with the issues raised?
- Which research methods have been used to investigate the proposed strategies?
- What gaps are there in current research?

To address these issues, a systematic literature review was conducted to collect evidence related to service composition in the form of a mapping study. Since "Web service" is widely used as a synonym for "software services" we restricted our study to this class of service. In the remainder of this chapter we describe how this was organised and performed, couple with reporting the results and providinge their classification.

3.2 Organisation of the Mapping Study

We based our study upon the guidelines from Kitchenham et al. (2007) and the reported experiences from Brereton et al. (2007). They studied those domains which are currently using systematic literature reviews, such as medicine and

education, in order to adapt the techniques for software engineering. Since the need was to identify the scope and form of existing experience, we have used a 'broad' form of systematic literature review, known as a 'mapping study' (Kitchenham et al., 2011).

The purpose of a mapping study (sometimes called a scoping review (Petticrew and Roberts, 2006)) was to identify the set of studies that addressed a topic and to categorise them. Such a review would address broader research questions than a full systematic literature review, with the aim to identify the 'gaps' (where primary studies are needed) and 'clusters' (where a systematic review might be applicable). The outcomes of a mapping study are usually in the forms of categorisation and counts of primary studies that do not normally involve aggregation of the outcomes from the primary studies.

The *review protocol* was used to specify the process that would be used to perform the review. In conducting our mapping study, the review protocol addressed the main activities associated with planning a review such as formulation of the research question, identification of the need for a review, research strategy, classification of papers, study selection criteria, study quality assessment and data extraction strategy and process. The following sections describe some key elements from the protocol.

3.2.1 Identification of the need for a review

Service-based technologies relatively have a short history (Brereton et al., 2007). Experience about what might be effective is also scattered across many sources. Since dynamic composition is an important (and technically challenging) feature of service technology, we wanted to find out what was known to performe this. The aim of the systematic review was to find as many primary studies as possible that contain empirically-based information related to service composition by using an unbiased and objective search strategy. The rigour of the search process is one factor that distinguishes systematic reviews from traditional reviews.

3.2.2 Search Strategy

To identify papers of interest, we needed to use keywords (index terms) that would capture the essence of the research topic. The research questions were aimed to provide the initial indicators to possible approaches and solutions that have been proposed. The scope of the review was defined by:

- *Population* is the published scientific literature reporting on Web service composition.
- *Intervention* is the different forms of service composition practices, techniques and processes.
- *Outcomes of relevance* are the quantity and type of evidence related to the effectiveness of various service composition techniques and processes.
- *Experimental design*, we were interested in any form of empirical study.

The search strategy is consisted of using a set of relevant keywords in order to search for primary studies and to identify the population, intervention and outcomes. The research strategy was to identify a set of electronic databases as principal sources of primary studies.It also involved to determine the relevance of papers found on the basis of the presence of particular key terms in the title and keywords list or abstract of a paper.

To search for related papers we prototyped the use of four strings: "Web service", "service composition", "Web service composition" and "dynamic Web service composition". We then decided to use the string 'Web service composition' as our search string, which includes what we looking for to answer the research questions. The advance search provided by search engines provides text boxes to enter our keywords and link them with the 'AND' operator to form the search string. In addition, we chose to search for this string in the "Title" and "Abstract" fields. To check the validity of our search string, we had identified a number of 'known' papers beforehand and checked that we found them in the outcome of using the chosen string.

In choosing our string we were aware of the fact that other terms were sometimes used for 'composition' (such as 'choreography' and 'orchestration'). However, it

is important to note that a mapping study does not seek to ensure as complete cover of the literature as a full systematic literature review of a more detailed topic. Our prototyping indicated that composition was sufficiently generic for our purposes.

We adapted our search strings to the interface provided by each search engine to seek papers that were fitted with our interests. Following the convention of limiting our search to a given full or half-year interval, we chose the interval 2000 – 2009 as we conducted our searching in 2010. The choice of 2000 was determined because in the same year the first draft of the SOAP standard was published.

3.2.3 Study Selection Criteria

The guidelines to conduct systematic literature reviews in software engineering recommend searching at least for four electronic databases (Kitchenham et al., 2007). We used three major archival databases:IEEE Xplore, ACM and ScienceDirect, and one indexing service (CiteSeer) on the pragmatic basis because these three databases and one indexing service encompass a wide range of sources for published papers relevant to service composition. In particular IEEE and ACM are major publishers of conference proceedings, which are particularly relevant for an emerging topic such as software service technology.

A systematic review of the literature should cover all potentially relevant resources. Therefore, we chose to include papers from conferences, journals and workshops. To avoid irrelevant papers we excluded literature that was only available as abstracts or in a slideshow format. Furthermore, we also excluded those studies that did not describe or make use of services composition and those that did not have an experience element, as well as excluding duplicates of papers that were found in more than one database.

3.2.4 Study Quality Assessment

The issue of the quality of individual primary studies is important for an SLR, where the objective was to synthesise the separate outcomes. For a mapping

study, which is mainly concerned with identifying "what exists?", it is common to omit any assessment of individual study quality. For this mapping study, we therefore included all the papers that were published in peer-reviewed journals and conferences, knowing that peer review has provided "good enough" quality assessment for our purposes.

However, in reporting in Section 4.5 (Chapter 4) on those papers that did report any outcomes or evaluation, we do indicate the extent to which the evaluation appeared to be rigorously conducted, without conducting any formal quality assessment.

3.2.5 Data Extraction Strategy

The mapping study process was designed to allow for the categorisation of published literature. Hence, the data extraction strategy was restricted to obtain only those information required for classification. For our review we needed to extract data relevant to the research questions. We sought to identify and record:

- The form of composition strategy.
- What were the issues raised or addressed.
- Whether the paper presented a potential solution and, if so, what is the nature of that solution.
- The research method used in the paper.

3.3 Summary

In this chapter, We introduced the motives that prompted us to conduct this research which were related to the emerge of Web service as method to build distributed systems over the Internet. We described the method as Web service composition. We identified the approaches to service composition that were currently being proposed. We introduced the systematic literature review

as means to perform our literature review (SLR) to cover the subject of this research. We proposed our research protocol to perform a SLR which is in mapping study form. Furthermore, we presented five steps of our research protocol with some details.

Chapter 4

Result From Systematic Literature Review

4.1 Introduction

This chapter presents the outcomes from conducting the Mapping study protocol. Here we describe how each step of the Mapping study has been implemented, and what results that have been found. This chapter aslo shows how the outcomes have been aggregated and synthesised using different form of classification. Furthermore this chapter identifies the strategies that have been used for dynamic service composition. The rest of the chapter illustrates a summary of papers on the dynamic Web service composition under these strategies.

4.2 Conduct of the Mapping Study

The conduct of the mapping study was the process of implementing the study design, as described in the previous chapter. This involved recording the outcome of the searches and noting any divergence from the design that may occur. Before starting to collect papers from each database, we familiarised ourselves with the relevant search engine's interface that we prototyped the search string which was used to query the specific database. The interfaces were somewhat

different, but they provide the same capabilities that were needed to run our query. Each interface has a box to enter the search string. We looked to find the search string in the title or the abstract of the paper. Moreover, there was an option about the publication year that we utilised where to extent possible to limit the search period to 2000 - 2009.

We created a form to record information about each paper we selected. Beside the standard citation data, the information included a description of the models and strategies, as well as the type of the composition (static or dynamic). We followed the pre-defined inclusion/exclusion criteria while searching the four databases. For a paper to be included in our set, it should present some kind of practical experience about WSC. Once this process was complete , we went through the outcomes again to ensure that all remaining papers would met our criteria. We also ensured to remove any duplicate entries from the final list.

4.3 Results of the Mapping Study

Our searching retrieved a moderately large number of relevant papers. After an initial filter we exculded those papers which were not relevant. The papers with duplicated entries in more than one database were also not included. Thus, finally we were left with 1172 papers. Table 4.1 shows the number of papers retrieved from each source (where there were duplicate entries, a paper is counted under the database that relates to its publication source). It was noted that most of the papers providing some form of experience were found in the IEEE and ACM archives. Applying the inclusion/exclusion criteria by using only the titles and abstracts of the papers reduced the number of papers to 206, which purported to provide some form of experience.

These papers were then classified by using the following attributes:

- The type of study (see Appendix A).
- Whether composition was dynamic, static or semi-dynamic.
- Publication type (conference, journal or workshop papers respectively).
- The research method used (see appendix A).

Electronic Database	Total Papers per Database	No of Experience Papers
IEEE Xplore	398	91
ACM	470	76
ScienceDirect	86	30
CiteSeer	218	9
Total	1172	206

Table 4.1: Total Papers per Database: these are the numbers obtained by using our keywords.

The technical reports by Kitchenham et al. (2007) and Brereton et al. (2006) provided valuable practical guidance to conduct the SLR. The basic analysis phase that consists of assigning each paper to one specific category, depends on the form of knowledge it presents. To perform this, we had to analyse each paper carefully, as some papers discussed more than one core issue, (e.g. a new technique combined with an experiment). In other words, a paper might be classified in different categories. For the purpose of this study we sought to put each paper only in one category. Categorization is useful technique to study the characteristics of Web Service Composition (WSC), as it offers a practical way to learn about each characteristic of WSC individually.

Wherever possible, we sought to classify papers by using the information provided in the title and abstract. Most papers explicitly mentioned the form of composition they addressed (static, dynamic or semi-dynamic) in their title or abstract. Many papers also provided information about the form of study, making it possible to classify them by research method too. However, for a proportion of papers, the poor quality of the abstracts required us to consult the full papers in order to decide their classification.

After applying the inclusion and exclusion technique criteria, the 206 remaining papers were classified into the different categories. At the same time some criteria to use in classifying these papers with regard to mechanisms commonly used in service composition were identified.

4.4 The Classification of the Papers

One of the goals of a mapping study is to categorise the papers to identify their major themes and the different methodologies that have been used in these papers to investigate the main theme. For this study we were interested in different composition strategies used (the themes) and the research methods adopted in these papers (Brereton et al., 2006). As indicated above, the initial classification into major categories was based upon the information provided in the titles and abstracts of papers. The fuller analysis provided in the following sections was based upon an analysis of the full contents of the selected papers.

4.4.1 Classification by Composition Method

For this, the experience papers were classified into three categories, as shown in Table 4.2, depending on the service composition models: that is, how these models address different application areas and requirements. These categories are enumerated as under:

- *Static* approach: This category contained papers that described or provided approaches explaining how Web service providers would offer their services to others. It also contained those approaches that need intervention from the user to develop or compose the service. Table 4.2 shows that most of the proposed approaches fall into this category.
- *Dynamic* approach: Forms of composition in this category involve the use of semantic descriptions of Web services. The process of selecting and controlling the final set of services (binding) takes place at 'runtime' without intervention from the users.
- *Semi-Dynamic* approach: For this the Web service developer is required to intervene for some stages of composition. However, the binding still takes place at run-time.

4.4.2 Classification by Research Method

Each paper was classified by the form of the research method and analysis used in the paper. This combined with the Web Service Composition strategy that was used to identify the data needed for answering the third research question in section 3.1. These research methods are listed in Table 4.2.

We employed five broad categories to classify the research methods adopted for the papers that we found. These categories are described as follows:

- *Framework/Conceptual Analysis.* Glass *et al.* have identified this as one of the major evaluation forms used in software engineering (Glass et al., 2004). Such papers usually employ an analytical framework to assess the ideas and models that they propose and may well not provide any actual implementation of these ideas.

- *MIS Tool/Conceptual Implementation.* , As observed by Glass *et al.*, this category is also a major form of evaluation used in software engineering, through which an idea is evaluated by constructing a tool or piece of software based upon it:"we built it and it worked".

- *Simulation.* This can be considered as a variation upon conceptual implementation. Here the implementation may be incomplete, that addresses only the key characteristics of interest byusing the simulation of the remaining parts.

- *Experiment.* This category includes those papers that have performed some form of controlled empirical study in order to evaluate the ideas or software concerned. Such studies are usually laboratory-based.

- *Case Study.* While an experiment seeks to isolate the object of study in order to investigate cause-effect relationships, a case study seeks to investigate it within its operational context (in the field) in exchange for having less control (Yin, 2003). So, this category includes those studies that investigate composition within a larger service-based environment.

The first three categories can be considered as being weak forms of evaluation, wheras the latter two are more rigorous (in different ways).

No:	Research Method	Static	Dynamic	Semi-Dynamic	No of papers
1	Framework/ Conceptual analysis	32	19	2	53
2	MIS tool/ Conceptual Implementation	45	52	7	104
3	Simulation	4	6	-	10
4	Experiment	14	10	-	24
5	Case Study	8	7	-	15
	Total	103	94	9	206

Table 4.2: Papers classified by research method and composition

In the following sections, we have chosen three categories from Table 4.2 (2, 3 and 4) to discuss in detail. In addition to this, we also discuss some of the papers from first category (Framework). Second category addresses the implementation of techniques that the developers have presented as solutions for services composition. The third and fourth categories from Table 4.2 were found to be very useful as mechanisms for testing the effectiveness of proposed frameworks. They helped in identifying most of the factors that involved in building a potential solution for service composition.

4.4.3 Other Classification Strategies

Although the practice that we adopted for classification (determining the set of categories through a mix of theoretical models and observation) is widely used for mapping studies, we should note the other strategies that have been suggested too. In particular, Wieringa and Heerkens (2006) propose classifying papers according to a conceptual scheme segregates papers into *design* or *research* categories, according to the type of problem that they address (*world problems* for which we seek a solution, and *knowledge problems* for which we seek information). They then use different criteria for the detailed classification of each type of paper.

Almost all of the papers selected for this survey are design papers, proposing ways of addressing the problems of composition. We therefore concluded that the more conventional categorisation described above was better suited to the set of research questions that we had identified. However, we also noted that the alternative scheme offered by Wieringa and Heerkens did reveal the lack of studies addressing *knowledge* issues.

Key Strategies	Total	Paper with Description only	Paper with an Evaluation
Semantic Web and Ontology	25	17	8
QoS (Quality of Service)	9	2	7
Knowledge-base and AI	6	3	3
Workflow and Composition languages	15	10	5
Middleware	7	4	3
Software Agent	6	3	3
Frameworks	12	0	12
Simulation Studies	6	0	6
*Others	8	6	2

Table 4.3: Key Strategies for Dynamic Web Service Composition
*Other approaches that we found were not related to one of the above strategies

4.5 Summary of the Dynamic Papers

Dynamic Web service composition (DWSC) involves performing binding (invoking) of Web Services (WS) 'on the fly' at run-time. The binding process occurs according to a composition plan which must be prepared in advance. A well planned DWSC should deal with the triad of service model elements (Provider, Customer and Registry). The plan comprises of a set of decisions about how to compose a service upon getting a request from a user that describs its intended business process through discovering potential published services from the registry in order to bind and execut the selected components (Fujii and Suda, 2005). Through our mapping study we have managed to identify a number of key strategies that have been widely adopted for building solutions for DWSC. The rest of the chapter discusses the papers that fall in the first six categories, as these embody a specific composition strategy (68 papers out of the 94 papers address the dynamic composition). Here we review the strategies which presented solutions related to DWSC and provided practical results (see Table 4.3).

The following sub-sections discuss the research strategies identified in Table 4.3. For the first six strategies, which map directly on to implementation of actual DWSC strategies, we provide tabular summaries of the papers concerned. If a paper provides an evaluation, we describe its form and outcomes, otherwise we only describe the form of implementation involved.

Figure 4.1 shows some of the information from Table 4.3, related to actual DWSC strategies, in a more visual format. The outer boundary line shows the

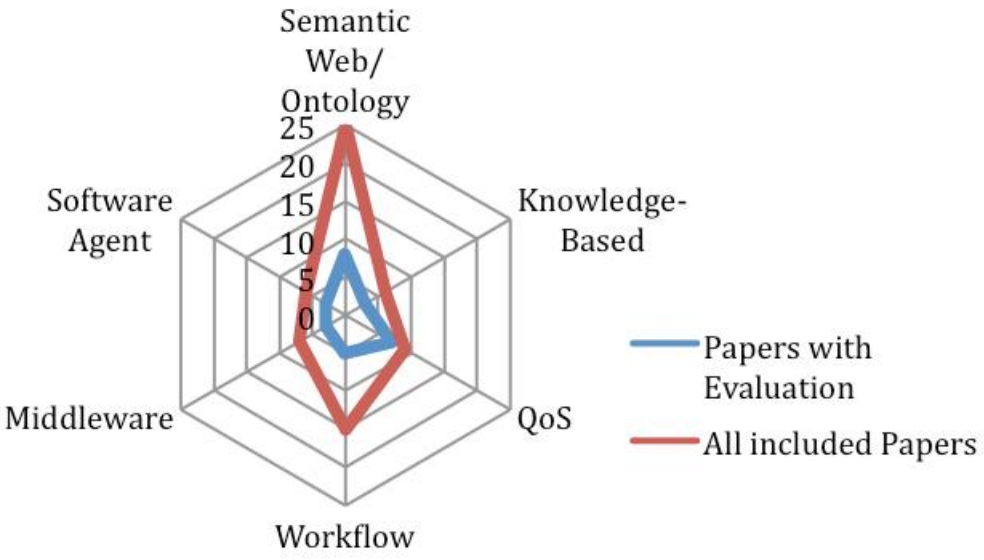


Figure 4.1: Radar-plot showing distribution of papers by implementation strategy

total number of papers describing some form of implementation of a strategy, while the inner line shows the number of papers that provide evaluations.

4.5.1 Semantic Web and Ontology

Accomplishing DWSC depends on many issues. Therefore, the detailed description of the WS itself should include functional properties, non-functional properties, preconditions, WS domain, among others . However, dynamic composition should occur without human intervention. This means the specifications of the Web services involved also have to be machine understandable. Thus, the semantics of Web service description is the most important element for enabling service composition to be achieved automatically (Zheng et al., 2008). Mechanisms such as the Semantic Web and related ontologies can assist with achieving it . User requirements can also be described semantically through description languages (Nie et al., 2006). This can assist with matching the service functions., Such languages may support both the functional properties as well as the non-functional characteristics for a WS. A description language partitions the semantic description into two components: a user pro-

file that describes the functions which the WS should supply through the use of the concepts defined in the Semantic Web ontology; and the domain constraint which states the qualitative preconditions that the Web services have to match.

Most of the papers addressing this strategy described languages that adopted experience from other IT areas to use in developing service composition systems, and employed a user specification to build the service composition model through different techniques. To support service composition semantically, different forms of languages have been proposed. Some of them have extended the XML-based languages for the purpose of enhancement or to add some needed feature, such as WSDL-S and OWL-S,enhancing WS description by adding semantics (Akkiraju et al., 2005). Moreover, some have made some adaptations to existing languages/tools to be used for modelling, constructing, verification or analysis of WSC. Examples of these are Petri Nets (Bing and Huaping, 2005) and UML-S, providing an extension to UML (Dumez et al., 2008).

As a comment about the literature addressing the semantic Web and use of an ontology, we noticed that most of the solutions approached DWSC in different ways.Nevertheless, all of these ways basically involve building around the user request profile that enriches the WS's description. Furthermore, those solutions claiming that semantic Web and ontologies would provide machine understandable techniques, make it possible for DWSC to be achieved automatically. For the approaches that fall under Semantic Web and Ontology strategy, Table 4.4 summarises the outcomes from the evaluations provided in each paper. Table A.2 (See Appendix A) summarises the outcomes of the approaches that provided sort of practical solutions for DWSC but with no formal evaluations.

Table 4.4: Evaluation of Semantic Web and Ontology strategies for Service Composition.

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes

Table: 4.4 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Cheung *et al.* (2004)	Studied the uncertainties involved in selecting and composing services. Used a General Environment that supports Grid/Web service composition. This environment uses the semantics of the available Web services (e.g., the semantics of the input/output parameters) as described by some machine understandable semantic Web language (e.g. OWL-S)	**Evaluation:** Ran two controlled experiments on the Globus platform (Grid computing). **Outcomes:** The results from the first experiment showed a significant reduction in service time with increasing numbers of nodes. The results from the second experiment have shown the effectiveness of the proposed bidding process.

Table: 4.4 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Medjahed *et al.* (2003)	Proposed a framework for composing Web services. Presented an algorithm to automatically generate composite services from high-level specifications of the desired composition. Defined a model for checking service composability. This model provides a set of composability rules that compare syntactic and semantic features of Web services. As an application domain of their research in Web services, they are partnered with the Family and Social Services Administration (FSSA) and used E-government as a case study.	**Evaluation:** Ran an experiment to assess the scalability of their approach, i.e., the possibility of generating plans for a large number of service interfaces. The aim was to evaluate the effectiveness and speed of the matchmaking algorithm. They also assessed the role of the selection phase in reducing the number of generated plans. They built a simulation testbed to run the experiments. **Outcomes:** The results showed that most of the time is spent on checking message composability. The execution time require comparing the parameters of each composite service operation with the parameters of each operation of a component service. With regard to generated plans, they particularly consider the composition completeness (CC) ratio. They conducted experiments for CC = 33% and CC = 66%. The results showed that the number of generated plans is higher for CC = 33%. Indeed, for CC = 33%, plans are generated if at least 33% of composer operations are composable with component operations.

Table: 4.4 Continued on Next Page. . .

Author(s)	**Composition Strategy**	**Summary of Evaluation & Outcomes**
Mokhtar *et al.* (2007)	Proposed COCOA as a solution to dynamic service composition in pervasive computing environments. COCOA provides COCOA-L, an OWL-S based language enabling the specification of service advertised and requested capabilities. Also provides two mechanisms: QoS-aware semantic service discovery and QoS-aware integration of service conversations.	**Evaluation:** Conducted a comparison between COCOA against the time spent for the XML parsing of services and task descriptions. **Outcomes:** Results showed that in more realistic cases, the overhead of using COCOA is negligible when compared to XML parsing.

Table: 4.4 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Ponnekanti and Fox (2004)	Examined four types of incompatibilities that may arise during interoperation: structural; value; encoding; and semantic. They address these incompatibilities using a three-fold approach: (1) static and dynamic analysis tools; (2) a GUI tool for resolving incompatibilities and generating cross-stubs; (3) a lightweight mechanism called multi-option types that enables applications to be authored. The implementation was primarily done within AXIS, an open source toolkit from Apache that implements static Java host types for WSDL services. Dynamic analysis and cross-stub generation was implemented by modifying the AXIS stub generators.	**Evaluation:** Performed three sets of experiments. (1) Application usage behaviour experiments to study what fraction of service functionality typical applications exercise. (2) Compatibility and integration experiments to demonstrate their tools and techniques in action. (3) Nature of incompatibilities experiments to study how often incompatibilities occur due to non-critical reasons. **Outcomes:** (1) The application owner has a choice of several non-native services to use, (s)he can automatically determine which among them are compatible using proposed techniques. (2) The results are based on CAT-S alone (compatibility analysis tool-static a tool that they provided in the paper and demonstrate the effectiveness of application usage behaviour inference in the integration process. (3) The results showed that applications typically exercise only a fraction of the service functionality.

Table: 4.4 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Segev and Toch (2009)	Proposed a Context-Based method for Matching and Ranking of WS. Studied three methods for classification of Web Services: Term Frequency/Inverse Document Frequency (TF/IDF); Context-based analysis; and a baseline method, which they used to extract information related to WS classification and ranking.	**Evaluation:** Used a set of experiments to compare their method with the other three methods mentioned. **Outcomes:** Showed that their method achieved better results than the string matching and TF/IDF method. In addition, they conducted other tests to compare the WS context that was extracted from WSDL and from textual descriptions and results show that the Web-based context extraction method analysing both the WSDL description and the textual description yields better results than the TF/IDF method and string matching.
Williams *et al.* (2003)	This paper showed how autonomous ontology merging for local consensus ontologies has potential for improving how agents conduct B2B Web service discovery and composition.	**Evaluation:** Conducted a number of experiments to evaluate their approach using syntactic similarity, semantic similarity, semantic relation discovery in terms of performance measures such as concepts merged, relations discovered, and computing time. **Outcomes:** The use of a lexical database increases the number of relationships found but increases the amount of time required to form the consensus ontologies. Also the experiment showed that rate of the number of concepts merged appears to decrease as the number of merge operations increases.

Table: 4.4 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Younas *et al.* (2006)	This paper devised to investigate into the efficiency of Web services composition. The authors presented a new protocol based on (I) P2P architecture, (II) active networks.	**Evaluation:** Used networks of queueing stations to model several scenarios in order to simulate the processing of Web services and message communication during the WS-composition process. Conducted three experiments in order to evaluate the efficiency of the proposed protocol (NetCom) in comparison to Self-Serv approach. (1) simulated the execution of a composite Web service composed from five component services (computes the mean response time). (2) conducted various experiments under different traffic loads to compute the response time of the NetCom protocol. (3) checked how the increase in the number of component services affects the response time of the NetCom 1&2 protocols. **Outcomes:** (1) The proposed NetCom protocol outperforms Self-Serv in response time. The NetCom protocol reduces the response time under different traffic loads. (2) The overall response time of the NetCom protocol is still better than Self-Serv. (3) Demonstrated that the NetCom 2 protocol performs better than NetCom 1 and Self-Serv approaches in the case of larger number of component services.

Table: 4.4 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Younas *et al.* (2005)	Proposed a SODA Workbench that includes a set of rules, knowledge (facts), and a reasoning mechanism to identify mismatches. The workbench contains a Distiller mechanism to extract WSDL and form a skeleton of VWS (virtual Web service).Used a case study of fast ferry concept design in order to test the proposed approach. A ship design process demonstrating the interoperability of different design services is presented, and the data flow between design teams in a distributed design environment is described.	**Evaluation:** A prototype system was implemented and tested for a ship design process. The experiment was carried out within a closed environment; that is prior to the experiment, the number of systems and their identity were pre-defined. **Outcomes:** Limited details of the outcomes, they just mentioned that the system is unable to provide support for choosing alternative services. It was suggested that in order to identify its full potential and limitations, future work would look at testing the system in an open environment.

Last part of Table: 4.4

4.5.2 QoS for Service Composition

Quality of service in this context means the ability to assign a different priority to applications, users, data flows or to guarantee a certain level of performance to a data flow (Tran, 2009). In terms of Web services, many approaches consider non-functional properties (NFPs) as general QoS properties. This includes any significant characteristics related to service consumers. Typical characteristics considered for QoS include performance, interoperability, reliability, accessibility, security, integrity, availability, etc. In some cases, different Web services might provide the same functionalities,therefore, QoS properties can have a significant impact on the expectations of service users that can be used to make a choice between similar services. Consequently, Web service descriptions should be strengthened by the addition of QoS attributes (Chaari et al., 2008).

QoS enhances DWSC in terms of guaranteeing the users a level of service quality

that meets their request. We found that the list of the elements involved in quality of Service is too long and have a variety of measurements. Consequently most proposed solutions adopted a different quality model in their approach, in order to make them applicable to their study. They do not make it easy to choose one solution that could be suitable for all DWSC, at least at the present time. For the approaches that fall under QoS for Service Composition strategy, Table 4.5 summarises the outcomes from the evaluations provided in each paper, and Table A.3 (See Appendix A) summarises the outcomes of the approaches that provided sort of practical solutions for DWSC but with no formal evaluations.

Table 4.5: Evaluation of QoS for Service Composition strategies.

Author(s)	**Composition Strategy**	**Summary of Evaluation & Outcomes**
Chen *et al.* (2006)	This paper showed how to discover and compose Web Services according to the input and output of service requests when there are a large number of services available. The mutual search operations among Web Service operations, inputs and outputs are studied, and a novel data structure called Double Parameter Inverted File (DuoParaInvertedFile) was proposed to implement these operations. Furthermore, they provided three algorithms to build DuoParaInvertedFile.	**Evaluation:** Implemented their algorithms on a test bed consisting of 640 Web Services and carried out a series of experiments to study the performance of the WS composition approach given in the paper. **Outcomes:** First, compared the approach based on three inverted files with their approach. They showed that their approach not only provided high performance but also low time growth rate. Second, compared time for Web services composition based on three inverted file versus that based on DuoParaInvertedFile. They showed that their approach is better than the former, and that the latter almost keeps steady when the level of WSCR (Web Service Composition Results) grows, showing that the more the number of service requests, the more effective the performance.

Table: 4.5 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Gu *et al.* (2004)	Presented an integrated P2P service composition framework called *SpiderNet*. The major contributions of this paper are summarized as follows. First, Spider-Net provides fully decentralized QoS-aware and resource efficient service composition using bounded composition probing. Second, SpiderNet provides proactive failure recovery to achieve failure resilient service composition. Third, SpiderNet achieves flexible service composition by supporting directed acyclic graph composition topologies and considering exchangeable composition orders to enhance the composed service's quality	**Evaluation:** Evaluated the performance of SpiderNet using both large-scale simulations and a prototype implementation evaluated in a wide-area network testbed, called PlanetLab. (1) In the large-scale simulation, they conducted an experiment to compare their algorithm with three other common approaches: optimal, random, and static algorithms. (2) They compared the QoS provisioning performance of SpiderNet with the random and optimal algorithms. **Outcomes:** (1)Their solution can achieve near-optimal performance with much lower overhead, and much better performance than random and static algorithms. (2) The average service delay of the service graphs discovered by the SpiderNet reduced with a growing probing budget. When the probing budget was very low, SpiderNet degenerated into the random algorithm, so the overhead was low, but the service quality was not satisfactory. SpiderNet could achieved near optimal performance with much lower overhead than the unbounded flooding scheme performing exhaustive searching.

Table: 4.5 Continued on Next Page. . .

Author(s)	**Composition Strategy**	**Summary of Evaluation & Outcomes**
Gu *et al.* (2003)	Proposed a QoS-assured Service composition framework (QUEST). Quality constraints such as availability and response time have been used to compose a qualified path under multiple QoS. Moreover, QUEST offers two re-composition algorithms which be used for a quick recovery when a service is not available or when the QoS has not been agreed.	**Evaluation:** Evaluated the performance of the initial and dynamic QoS-assured service composition algorithms using extensive simulations. The simulation reports about the violation rates of two QoS attributes: availability and response time, respectively. **Outcomes:** The results showed that QUEST can provide both QoS assurances and load balancing for composed services in SON (Service Overlay Network). The simulation results also indicate that the partial dynamic service re-composition algorithm can achieve almost the same level of QoS assurance as the complete re-composition algorithm, but with much lower overhead.

Table: 4.5 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Liu *et al.* (2004a)	Presented an extensible QoS model that is open, fair and dynamic for both service requesters and service providers. This achieved the dynamic and fair computation of QoS values of Web services through a secure active user's feedback and active monitoring.	**Evaluation:** Conducted a series of experiments to: (1) investigate the relationship between QoS value and the business criteria; (2) study the effectiveness of price and the service sensitivity factors in their QoS computation. They simulated 600 users searching for services, consuming the services and providing feedbacks to update the QoS registry in the UPS application. **Outcomes:** (1) Showed how selecting Web service depends on user quality preference when more than one Web service matching the required functionality. (2) The result showed that using the the same sensitivity value for both price and service factors would not be effective for a price sensitivity search in a QoS registry.
Sun (2004)	Addressed a QoS composition reasoning approach to predict the system QoS based on the QoS of individual components. The proposed QoS composition reasoning approach consists of a QoS composition meta-model, QoS composition model and QoS artefacts from a business specific profile, architecture specific profile and technology specific profile.	**Evaluation:** Performed experiments to validate the proposed composition rules of throughput and turnaround time. The experiments were conducted in an environment where there were fixed number of clients and each client sent a certain numbers of requests with a uniform distribution. **Outcomes:** Claimed that the proposed composition rules accurately predicted the throughput and response time of a composed system of two single threaded components communicating using synchronous two-way invocation or asynchronous two-way invocation.

Table: 4.5 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Yan *et al.* (2007)	The service level agreements for a service composition were established through autonomous agent negotiation. To enable this, a framework is proposed in which the service consumer is represented by a set of agents who negotiate quality of service constraints with the service providers for various services in the composition. This negotiation was well coordinated in order to achieve end-to-end quality of service requirements.	**Evaluation:** Implemented a prototype that innovatively exploits the agent technology to address the QoS issue and SLA. Actually their evaluation is a use case rather than a case study as they claim, because they used it as an example to show the capabilities of their framework. **Outcomes:** Used their prototype to simulate the SLA negotiation through a pre-defined scenario. The result showed that the outcome of the negotiation process has chosen the best offers of QoS.
Zeng *et al.* (2003)	Proposed an approach based on the QoS model using linear programming techniques to compute optimal execution plans for composite services. Discussed the relationships between their work (Quality driven Web services composition) and existing Web service standards, Web service composition approaches, and QoS-driven Workflow management.	**Evaluation:** Conducted experiments to compare the proposed technique with the local selection approach. **Outcomes:** The results showed that the proposed approach effectively selects high quality execution plans (i.e., plans which have higher overall QoS).

Last part of Table: 4.5

4.5.3 Knowledge-based and Artificial Intelligence (AI)

In this context, the term 'Knowledge-based' implies storing knowledge about services in a computer-readable form, usually for ontological or semantic purposes, to assist the tasks related to service composition such as discovering or

selecting a Web service. Having essential and adequate knowledge as well as practical experience available in this way, can be considered as important intellectual assets that help in making optimal and practical engineering decisions. This knowledge has to be clearly recorded, saved and reused (Wang and Taylor, 2008).

The papers that have used knowledge-based approaches, employ the same strategy of involving domain knowledge to develope enough shared vocabulary to assist the service selection process (Xiaogao and Xiaopeng, 2006, Jihie et al., 2004). Moreover, they use the shared vocabulary to give a basic description of the service (component), such as the input and output parameters.

Both knowledge-based and AI approaches have been used to support the ontological and semantic purposes. The difference is in the usage of intelligence to automate the DWSC processes through computer-readable form. Although it offers a promising solution to DWSC, it is hard to construct. That is because they need experts to build them. For the approaches that fall under Knowledge-based and AI forms for service composition strategy, Table 4.6 summarises the outcomes from the evaluations provided in each paper, and Table A.4 (See Appendix A) summarises the outcomes of the approaches that provided the sort of practical solutions for DWSC, without containing any formal evaluations.

Table 4.6: Using Knowledge-based forms for service composition.

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes

Table: 4.6 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Kim *et al.* (2004)	Used a knowledge-based approach to describe the components and the input and output that could be used in a Workflow. Using a knowledge-based technique provides rich representations of components together with planning techniques that can track the relations and constraints among individual steps.	**Evaluation:** Illustrated their approach by implementing a system called *CAT* (Composition Analysis Tool) that analyses workflows and generates error messages and suggestions in order to help users compose complete and consistent workflows. **Outcomes:** Used examples to demonstrate the potential of the system, with the tool being applied to two different applications: constructing workflows in the travel planning domain and constructing computational workflows in the earthquake science domain.
Madhusudan and Uttamsingh (2006)	Presented an AI-planning based declarative service composition approach, called *Integrated Service Planning and Execution* (ISP&E) for building robust on-the-fly customized Web services in dynamic environments.	**Evaluation:** Conducted a set of experiments to evaluate the advantages of interleaving planning for service composition and execution in providing reactive behaviour over a static hardwired service composition approach. **Outcomes:** The strategy of interleaving planning and execution was considered viable when the dynamics of the environment are moderate. If the rate of change is high, it may be cost-effective to plan and execute dynamically. Overall for all strategies, as service request size increases, the total planning time increases as the number of operators in the plan increase.

Table: 4.6 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Yu and Yu (2006)	Proposed a knowledge-Based advice system which is aimed at employing semantic service descriptions. Also they discussed using a knowledge-based approach for resource synthesis and to provide advice on service selection and instantiation. Their work emphasized the importance of DAML-S, and related technologies, in providing semantically-enriched characterizations of available services.	**Evaluation:** Implemented a prototype for the Workflow construction environment that supported the runtime recommendation of discovery and select processes. Outlined a service-oriented architecture for knowledge based systems operating in the context of the technological infrastructure provided by Grid-computing platforms and the semantic Web. **Outcomes:** Using the prototype, they outlined the importance of domain knowledge with respect to just one aspect of expertise, namely the selection and configuration of services as part of a Workflow specification. Not really an outcome, the author said "the full evaluation of this system awaits further investigation and user feedback" .

Last part of Table: 4.6

4.5.4 Workflow and Languages for Service Composition

A Workflow is a concept commonly used for describing a business process. The specification of a Workflow consists of fundamental component structures (Lopes et al., 2008). Furthermore, a Workflow can be used as a modelling tool to represent real work for further assessment. For Web services, a Workflow can describe a network of service operations, with the communication between them would be through the data links that represent the relation of the outputs of some operations to the inputs of others (Belhajjame et al., 2008).

Contributors have tackled the composition flow from two views; control-flow (process order) and data-flow (message passing), in order to manage the service composition.

Approaches that fall into this category have presented different technologies

and solutions towards DWSC. Most of them work to simplify the composition processes for the user through form of modelling, visualization and ready building blocks (templates). The simplicity they offer makes it easy for users to map their requests to a composition plan. For the approaches that fall under Workflow and Languages for service composition strategy, Table 4.7 summarises the outcomes from the evaluations provided in each paper, whilst table A.5 (See Appendix A) summarises the outcomes of the approaches that provided practical solutions for DWSC but with formal evaluations.

Table 4.7: Using Workflow and language forms for service composition.

Author(s)	**Composition Strategy**	**Summary of Evaluation & Outcomes**
Ambite and Weathers (2005)	Described work towards an approach to automatically generate computational workflows for transportation modelling problems. The work relies on ontology of the application domain to provide formal semantics to the sources and operations available. It describes the ontology, the sources, and operations as a Triple logic program. This program generates workflows to answer user requests. It focused on workflows with aggregation operations.	**Evaluation:** Designed two experiments to test the scalability of the Workflow generation program. (1) Experiment tested the ability of the program to find a solution Workflow in the presence of a large number of sources, out of which only a small fraction provide answers to the user request. (2) Experiment tested how increasing depth of the hierarchy affects the Workflow generation. **Outcomes:** (1) The current system could generate workflows with up to 10000 additional sources (30000 descriptors) in less than 45 seconds. (2) The system took up to 108 seconds to compute a Workflow with 256 sources and 511 operations.

Table: 4.7 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Chafle *et al.* (2004)	This work covered issues ranging from code partitioning for decentralization to detailed discussion of the servers that participate in decentralized execution - their thread pool design and communication protocols. Also discussed build and runtime issues in error handling and error recovery	**Evaluation:** Experiments were conducted to study the performance of centralized and decentralized orchestrations using the *FindRoute* example discussed in their paper. Dummy implementations were used for services. Used the BPWS4J engine to orchestrate specifications written in BPEL4WS. Used two multi-threaded asynchronous clients from two different machines to load the system. **Outcomes:** Observed that decentralization provides performance benefits even in cases where there was no inherent concurrency. The performance gain was mainly due to reduction of network traffic and distribution of computation across different nodes. Also, observed that although horizontal scaling and load balancing helps centralized orchestration perform better than decentralized orchestration at low loads, decentralized orchestration scales better at higher loads. This can again be attributed to optimal utilization of threads in a decentralized orchestration system, as they are not blocked waiting for responses and allow large number of concurrent requests to be executed.

Table: 4.7 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Chi and Lee (2008)	Proposed the SCMP platform as an MIS tool to provide visual modelling, reliable Workflow measuring and composition script translation mechanisms. SCMP is a formal modelling platform which adopts the control flow and capabilities of Petri Nets components to generate the Workflow. They have created an implementation of their platform.	**Evaluation:** Used a Loan broker example to demonstrate their platform mechanisms. The loan broker process was denoted in terms of Petri Nets used to help the developers to conceive interrelated service components according to process logic, and then create a conceptual model using the visual editor. Used PNML for reliability measurement programs, an implementation independent language used to produce machine-executable code from the conceptual model. **Outcomes:** Claimed that the script language translation helped developers to generate code from the visual composition. Furthermore, SCMP could connect proper Workflow engines to implement runtime from the compositions. Most importantly, the measurement mechanism demonstrated the composition reliability by validating a translated Petri Nets model.

Table: 4.7 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Fu *et al.* (2006)	Presented the basic idea of the SO-SAM model. Claimed that the Software architecture of a Web services-oriented system can be described by building an executable SO-SAM model. Introduced a case study used throughout the paper. It deals with a simplified online shopping example adapted from their previous work. Its a basic electronic commerce process of online shopping with credit card transactions.	**Evaluation:** Used the Maude[1] model checker to verify the property specification in SO-SAM model. To automatically implement the model checking using Maude, they designed an algorithm to translate SO-SAM (also SAM) to the Maude programming language. **Outcomes:** Claimed their modelling and validating approach helped to enhance the reliability of Web service-oriented applications.
Nanda *et al. (2004)*	Devised a new code partitioning algorithm that is applicable to decentralization of composite Web services. The algorithm depends on a technique for testing for legality of reordering of PDG (program dependence graph) nodes, and on a technique to estimate the throughput of a network of servers executing a business process.	**Evaluation:** Conducted experiments to study decentralized orchestration of Composite Web Services. Ran these experiments to test the Runtime Performance and Compile-time Performance, using a number of examples. **Outcomes:** Their experimental results showed that decentralization could increase the throughput of composite services substantially, easily doubling it under high system load.

Last part of Table: 4.7

[1] Maude is a freely distributed high-performance system, supporting both rewriting logic and membership equational logic

4.5.5 Middleware for Service Composition

Middleware is a computer software that sits between applications software and a range of different platforms. Middleware is used to make the interaction between different applications possible, including exchanging data, message passing, development, executing some task, etc. Its use hides the complexity of the systems and the network details from the users in order to facilitate the service composition processes. Additionally the use of a Middleware system provides support for distribution, heterogeneity, interoperability and mobility (Ibrahim and Le Mouël, 2009).

Adapting middleware for services composition can support the integration of heterogeneous services in small networks such as Home Area Network (HANs).

The usage of middleware has taken care of the most difficult part of the composition processes, such as the integration. This allows the developer to be more focused on the composition logic rather than technical issues. Although middleware can do much better,we found that it was still being used in small scale, especially within small networks such as home or computer labs. For the approaches that fall under Middleware for service composition strategy, Table 4.8 summarises the outcomes from the evaluations provided in each paper. Table A.6 (See Appendix A) on the other hand summarises the outcomes of the approaches that provided practical solutions for DWSC but with no formal evaluations.

Table 4.8: Using Middleware for Service Composition.

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes

Table: 4.8 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Chakraborty *et al.* (2004)	Proposed a distributed broker-based composition protocol, and compared the protocol to a Fixed-node based Composition protocol where all the requests are sent to a preconfigured node in the system.	**Evaluation:** Implemented on an ad-hoc network simulator (Glomosim) under various service densities, mobility and topologies. Compared the protocol to the often-used Fixed-Source based Composition protocol where all the composite requests generated in the system are sent to one fixed node that acts as the composition engine. Ran an experiment by using a simulation over a set of 64 nodes, following random way-point mobility with speed of 2m/s and stoppage time of 5s. **Outcomes:** Claimed that their protocol performs better than Fixed-Source based Composition protocol in locating nearby nodes that contain the required services. This is largely due to the topology-sensitive placement of the Broker in the Broker Arbitration Phase.

Table: 4.8 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Pourreza and Graham (2006)	Proposed a middleware for local interaction environments. This approach offers several advantages, such as the service provider performing the bulk of the composition work, Providing new services “on the fly" as devices are added to a HAN, also makes their use more attractive to technically unsophisticated home owners. The middleware mitigates maintenance problems. The Workflow created for one home will be added to its repository, thereby reusing compositions to reduce future composition overhead. Furthermore, using existing in-home protocols provides seamless integration of new composite services with existing ones.	**Evaluation:** A prototype has been used to study service composition in HANs (home area networks). that is also applicable to other local interaction environments such as classrooms and conference halls. **Outcomes:** The prototype has been used to explain the functionality of their framework. Claimed that they have tested the system using a number of emulated UPnP devices and several sample Jini services which are fully functional, but no details of these are provided.

Table: 4.8 Continued on Next Page. . .

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Williams *et al.* (2005)	The current implementation architecture of their middleware for WSC in pervasive computing is centralised, as they believe that the pervasive environment will have centralised controllers for each local context. The approach acknowledges that agents may have diverse ontologies which may have impeded their ability to relate capabilities at a slightly higher level. Therefore, in this approach, agents create relatively small local consensus ontologies to facilitate discovery and understanding.	**Evaluation:** Ran a number of experiments. The first experiments focus on measuring concepts as merged/learnt or discovered relation/similar ontologies to reach consensus. The second experiment followed the same steps of the first experiment but using a Lexical database to study the benefit and limitation of using a Lexical database. **Outcomes:** The observations from the first experiment were: (1) The addition of new concepts declines as the number of ontologies merged increases; (2) The number of concepts/relations discovered per merge remains relatively the same throughout the experiment; (3) As the percentage of similar concepts increases, the number of new concepts/relations learned decreased. The observations from the second experiment were: (1) Early operational performance (semantic matches) increases are realized using the lexical database; (2) Using the lexical database, the number of new relationships is greater than without throughout; (3) Considering relative performance (service time), the lexical database is not as effective when the number of same or similar concepts is high.

Last part of Table: 4.8

4.5.6 Software Agent for Service Composition

A software agent can be an autonomous or at least semi-autonomous entity, which can execute tasks to achieve a specified goal without intervention or with minimum supervision or direct control. Due to the complexity of the WSC process (WS are distributed, dynamic and heterogeneous) that involves many tasks, such as discovering, selecting, binding the services and controlling, the Workflow of the composition agents can perform one of these tasks on behalf of the user in order to make the composition processes less arduous. The behaviour of agents depends on the composition context, which means any information relevant to describe the situation is adopted. In the course of composition, a software agent undertakes negotiation with their peers to choose the Web services that is used for the composition (Maamar *et al.*, 2004).

Some of the service composition approaches benefit from the ability of agent software to integrate and connect services to achieve a business goal.

The approaches that used Software agent strategy for DWSC has employed its flexibility to cope with DWSC in different ways, e.g., discovery, negotiation, fault tolerance, etc.. Like the knowledge-based and AI strategy, it is not easy to construct. However, when it is built then it can do the difficult part on behalf of the developer. For the approaches that fall under Agent software for service composition strategy, Table 4.9 summarises the outcomes from the evaluations provided in each paper. Likewise, table A.7 (See Appendix A) summarises the outcomes of the approaches that provided practical solutions for DWSC without describing the formal evaluations.

4.5.7 Frameworks for Service Composition

A framework is a basic conceptual structure used to support reuse when solving or addressing complex issues of DWSC. Therefore, it can propose mature and comprehensive solutions. Furthermore, these frameworks have common features, since each of them includes diverse components such as code libraries, a scripting language, support programs, or other software to assist developing and linking the different components of a service composition. Moreover, they use more than one DWSC strategies, we already been mentioned in this chapter.

Table 4.9: Software Agent for Service Composition

Author(s)	Composition Strategy	Summary of Evaluation & Outcomes
Blake and Gomaa (2005)	Presented *Workflow Automation for Agent-Based Reflective Processes* (WARP) as a standard software engineering process and a language that supports the specification of the Workflow processes and control.	**Evaluation:** A prototype has been implemented. Conducted an experiment by using this to evaluate the impact of the overhead caused by the three modes of real-time configuration (presented in their paper). **Outcomes:** Claimed that the increase in overhead by operational mode was anticipated. Another result was that the overhead increased with the increase of Workflow steps. This was relatively unexpected since the latency was constant in the base case as the number of Workflow steps was increased.
Jayaputera *et al.* (2007)	Presented the notions of mission-based MAS (Multi-agent systems) and on-demand agent generation. Also a formal model of the mission and TDG (task decomposition graph) as well as mission execution and run-time plan modification. Presents case studies which they used to evaluate their design concept and implementation of the *eHermes* system.	**Evaluation:** Conducted a number of experiments to observe and study their approach when operating under different situations and workloads. **Outcomes:** Concluded that the *eHermes* system is capable of handling complex missions that can contain as many as 1500 tasks. Despite such a large number of tasks, eHermes can generate agents efficiently and handle the workload adequately. However, they noted that to fully test and demonstrate *eHermes* capability, complex real world scenarios and data sets are required.
Maamar *et al.* (2004)	Proposed a Web Services Composition Approach based on Software Agents and Context. The approach employs different types of software agent and three levels of specification. The different agents have been aware of the context of their respective services in the objective to devise composite services on-the-fly. Also presented a specification approach for Web services composition and deployment.	**Evaluation:** A prototype has been implemented. The prototype architecture consisted of a service composition environment and a pool of services and agents. The service composition environment consists of a set of integrated tools that allow service providers and users to create and execute services. WSDL is used to specify Web services and UDDI is used as a service repository. The approach has been characterised by the use of different types of software agents and three levels of specification. The levels were denoted by intrinsic, organizational/functional, and behaviour, and illustrated with a running scenario. **Outcomes:** Little detail is provided, they mentioned that their ongoing work will provide an assessment of the performance and scalability of the proposed agent-based multi-domain architecture.

We identified ten papers that fall this category. The use of frameworks in these papers is summarised in Table 4.10. The columns in Table 4.10 describe the set of features that characterise these frameworks, as extracted from the set of papers. Broadly, the features can be considered as architectural, e.g. such as Agent-based); descriptive, i.e. Visual/GUI; or concerned with creating a solution, e.g. support for use of natural language. The most obvious exception is fault-tolerant which is more related to a specific goal for one framework. Thus, while each framework has only one architectural form (where this is considered to be a relevant feature), it may use multiple means for the other elements.

Table 4.10: A comparison between Service composition framework papers.

Frameworks	Visual/GUI	StatementChart	PBD/Data Flow/Graph	Semantic/Ontology	Agent	OO	template/Form	Support Natural Language	Support select() operation	Fault-Tolerant
(Deng *et al.*, 2004) ASCEND	✓	✓	✓	✓					✓	
(Patil *et al.*, 2004) METEOR-S	✓		✓	✓					✓	
(Lee *et al.*, 2005) Canal				✓	✓				✓	
(Fujii & Suda, 2004a) CoSMoS & SoGSoC	✓			✓		✓		✓		
(Braem *et al.*, 2006) SCE	✓		✓				✓			
(Orriëns *et al.*, 2003) ServiceCom	✓		✓						✓	
(Hall & Cervantes, 2003) Gravity	✓		✓	✓						
(Schmid *et al.*, 2006) LARA++	✓		✓	✓			✓		✓	
(Agarwal *et al.*, 2004) OntoMat	✓		✓	✓						
(Liang *et al.*, 2007) DwO				✓		✓				
(Benatallah *et al.*, 2005) Self-Serv	✓	✓	✓	✓	✓				✓	
(Liu & Mingjun, 2010) FACTS	✓		✓	✓						✓

The analysis in Table 4.10 indicates the main features with different frameworks sharing visualization/GUI, graphs, Web Service semantic and Ontology and the basic composition operations. It also can be considered as the intersection of the Web service composition strategies. The visualization and GUI elements

are employed in most composition frameworks to help the user in understanding and deal with design issues, and to develop the composition model. Graphs and diagrams are factors that sustain the efficiency of the proposed approaches by offering a number of shapes that make the concepts of what is being composed easy to understand and manipulate. Another factor is the use of semantics and ontologies when this factor is fully developed it should be able to improve the composition process constantly and make it better able to achieve the intended goals. The last factor is the provision of certain composition operations which can offer a comprehensive composition platform to solve a user's business issues through interactions with other businesses processes or customers.

4.5.8 Simulation

Simulations have been used for off-line analysis to study systems design, as well as to explain the effects of change and redesign of operating policies. Where reliable statistical analysis tools are available, simulation technology may be beneficial to use for real-time embedded control of large-scale systems. In this case, metrics of performance depends on data obtained in both temporary and steady-state periods (via simulation) That may be used for the development of real-time control policies to deal with a large-scale system effectively (Law, 2008).

We have found that simulation in service composition approaches has been employed in different ways to support the development of service composition systems.

- Pfahl (2007) has identified the likely benefit that simulation can offer for services composition such as improvement of technique and tools, standardised representation in collecting the empirical evidence and improved knowledge transfer.
- The distributed simulation, proposed by Chen *et al.* (2006), relies on components files to execute the simulation description file of the clients as part of distributed simulation application development in the service-oriented architecture.

- WPDT, proposed by Chandrasekaran *et al.* (2002), employs the combination between simulation and Web process as a technique to improve/correct the design of Web processes. They map the flow described in an WSFL model to a JSIM[2] model to simulate the behaviour of composed Web services.

- The framework of Tsai *et al.* (2006) is service-oriented, agent-based and discrete-event driven. Application tasks are represented as services or agents that interact according to scenarios defined by the specification and modelling language.

- Madhusudan and Son (2005) study's shows the adaptation of online simulation for scheduling service execution and operating policies. In addition, the study presents the real-time simulation in the ISP&E to guide process execution which is used as essential part of the decision-making process.

- Madhusudan and Uttamsingh (2006) investigate the ISP&E approach for dealing with the nature of Web services. Their experiments studied the planning and execution time under different strategies through use of a prototype. Another experiment shows how embedded simulation framework and guidance can enhance the time of re-planning (and re-execution) when the failure of an individual Web service would occur.

Obviously there are various strategies for DWSC, even though the studies within same strategies showed different techniques. In turn, the simulation literature has reflected different proposals. Although we did not find a simulation study that could be applied for all DWSC strategies, it can help the developers to evaluate their particular proposal under its particular situation.

4.6 Summary

As emphasised at the start of Chapter 1, we first set out to conduct a *mapping study* for which the aim was to find out how much of the published literature

[2]a Java-based simulation and animation environment being developed at the University of Georgia

about WSC is actually supported by experience of some form, where this might be based upon implementation, experiments, prototyping etc. Based upon this we planned to identify promising forms to study further. The mapping study showed that approximately one in six of the studies provided actual experience, which suggests that the literature regarding composition is probably still largely based upon modelling rather than experience.

With regard to the specific research questions that we asked.

- *What are the main issues that need to be addressed if dynamic service composition is to be successfully implemented and widely adopted?* Over a third of papers addressed dynamic service composition, and used a wide variety of approaches. The difficulties found in comparing these showed that there is an urgent need for an agreed set of benchmark scenarios that describe dynamic composition of a set of services in order to help with assessing how well different composition strategies work, and in what situations they work best. Clearly, the incorporation of semantic knowledge into the composition process is another key issue too, but beyond this, it is simply not possible to identify any clear patterns related to ideas about composition at this stage in its evaluation.

- *What solutions have been proposed to deal with the issues raised?* The solutions we found in our survey have been outlined in Chapter 4. Arguably, this preoccupation with world problems rather than with knowledge problems is actually an impediment to progress. At this point in time, we need a deeper understanding of the nature of service composition (and some good examples), rather than any additional design solutions.

- *Which research methods have been used to investigate proposed solutions?* Table 4.2 (see chapter 4) shows that the largest category is that of conceptual implementation (which is commonly the case for software engineering knowledge). The second largest category was that of frameworks, with more empirical forms making up the rest.

- *What gaps are there in current research?* The very scattered nature of research into this topic and the lack of district 'clusters' of studies makes

it hard to really identify meaningful gaps. When we posed this question, we did expect rather more experience to be found than proved to be the case, and so we cannot really answer this question on the basis of the outcomes from this study.

While this is perhaps not a surprising set of answers, there is at least one obvious conclusion that we can draw from this. That is, conferences and journals do need to emphasise the need for authors to include evidence about the effectiveness of a strategy wherever possible. The problem we face is not a lack of models, but a shortage of studies that seek to apply those models in some way and to assess how effective this is.

A mapping study can sometimes form the basis for a subsequent systematic review, by identifying where there are usable 'clusters' of empirical results that might be aggregated in some way. However, if we look at the six groupings of interest, we find the following.

Semantic Web and Ontology. The experimental studies for this forms have largely focused on either measuring the overheads imposed by the chosen approach, especially in terms of time and the scalability of the approach, or upon assessing the 'effectiveness' of service selection. The variety involved and the lack of common measures provides little opportunity to aggregate these outcomes in any way.

QoS. Again there were a number of different approaches proposed and studied, but no overlap or comparison between them. However, results from the studies were generally positive across the measures used, suggesting that this approach merits further study.

Knowledge-based and AI. Only three studies provided evaluations, and one of these was rudimentary.So they provided no scope for aggregation.

Workflow. Three studies looked at performance, while two examined the reliability of the composition process.

Middleware. There was no common basis for the small set of evaluations provided.

Agents. The small set of outcomes were mainly concerned with performance.

Taken as a whole, where an evaluation was performed the papers seemed to report on either:

- Performance of the system (particularly any overheads)
- The quality of the outcomes (usually in terms of whether an 'optimum' set of services were selected)

However the lack of comparative studies or of the use of any common baseline (benchmark use cases) means that outcomes are essentially local in scope. A few studies used the 'classical' composition problems (such as travel planning).

From this, we can identify two steps that might help provide a much stronger basis for empirical knowledge in this area.

- Agreement on a set of common measures (performance, quality of solution etc.) that might aid comparison between approaches.
- Establishment of a set of agreed 'benchmark' use cases and related scenarios that can be used by experimenters. The analogy that we have in mind is with the 2007 International Timetabling Competition[3] where the organisers provided a 'use case' in the form of a set of timetable data and constraints that were based on those for a real university. Entrants to the competition can then use this data-sets with their preferred technique. The results in the form of timetables and the lists of any constraints that fail to meet can be compared and published.

We would argue that adoption of these ideas by the community greatly enhances the value of any empirical studies, and also encourages greater use of these (remembering that the set is still very small when compared to the larger set of papers about Web service composition).

[3] – http://www.cs.qub.ac.uk/itc2007/index.htm

The outcomes from the mapping study help in identifying the major themes and the different methods that have been used for Web service composition. By employing the mapping study, we are able to form a generic life-cycle model for Web service composition. This model contains essential processes that can be adopted by researchers and practitioners to develop their own solutions for service composition. Furthermore, we use this model to study the 29 empirical papers with details (see Chapter 5). The outcomes from the mapping study help in identifying two strategies for service composition; which are Ranking and QoS. We study these two strategies through a replication and comparative analysis. The study based on building a simulation framework to carry out some experiments on these two composition strategies.

Chapter 5

Modelling the Process of Composition for Software Services

5.1 Introduction

SOA provides a conceptual framework for building distributed systems (Erl, 2004, Potts and Kopack, 2003). The framework has three basic components: Service Registry, Service Provider and; Service Requester. In addition to this, this framewok has three essential operations: Publish, Find and Bind), details are given in Chapter 2, Section 2.3. Therefore, all approaches adopting the SOA framework need to build their solutions around these operations.

The practical applications of the SOA framework for building distributed systems (such as composite Web services) show that additional support operations are required for performing the steps of service composition (Yu et al., 2007, Badr et al., 2008, Rong et al., 2008). The purpose of these additional support operations is to describe an exhaustive service composition life-cycle which should cover the entire service composition process.For example, the identification of user requirement and the development of a composite service plan and its execution. The studies that have proposed specific Web service composition life-cycle models (Yang et al., 2003, Aslam et al., 2007, Chhetri et al., 2007, Rong et al., 2008) are summarised in Table 5.1 and Figure 5.1. These life-cycle models are broadly similar in terms of their concepts and the processing order

that they propose for the Web service composition process However, they are different in terms of performing their steps. Each of these life-cycle models consists of phases/sub-cycles that deal with some stage in processing a user request.

For Web Service Composition (WSC), the specification languages in use are: BPEL4WS (IBM, 2007, Chafle et al., 2004), WSFL (Leymann, 2008), WSCI (W3C, 2002), etc. These are XML-based languages that can be used for the specification of a composite Web service, where it is interpreted/executed by an engine, such as BPWS4J (IBM, 2002). Generally, a composite Web service specification is interpreted by a single coordinator node (the service composer), as described by (Chafle et al., 2004, Dustdar and Schreiner, 2005), in order to create the composite service. The *find()* and *bind()* operations are the essential elements of the Web service composition task. The service composer uses the *find()* operation to enquire about any WSs that can fulfil a specific task, which are available from the registry. Afterward it chooses a specific WS according to the Service Requester's (consumer) requirements, and uses the *bind()* operation to interact with the WS providers by sending a request to WS components that take part in the composition. These WSC models have also expanded the SOA framework by developing additional support operations for building a complete framework for Web Service Composition.

5.2 A Generic Life-Cycle Model for Web Service Composition

From the literature review conducted in this research,the four papers mentioned in Table 5.1 are ones that have proposed life-cycle models for service composition. In these models different activities (supportive operations) are used to develop a life-cycle model for Web service composition. In order to create a generic model from these, we followed the process described below:

1. Firstly, we classified these activities into four groups that represent the basic activities involved in Web service composition life-cycle. These groups are identified from the summary of the four Web service composition

		(Yang et al., 2003)	(Aslam et al., 2007)	(Chhetri et al., 2007)	(Rong et al., 2008)
Web Service Composition Life-cycle	User requirements	**Definition** (abstract definition for the composite service)	**Business process modelling**		**Service-oriented function description** (user's requirement)
	Find/Discovery and selection	**Scheduling** (developing & assessing the composite service)	**Development**	**Planning** (discover and compose services to meet the user's goal)	**Service design and implementation** (mapping the requirement into detailed plan)
				Binding (negotiation and agreement sub-cycle uses QoS)	**Service composition** (discover, reference & bind certain service, plus creation & classification of new services)
				Enactment: SLA & monitoring (SLA formulation, monitors the actual QoS during service enactment)	
	Workflow development & Execution	**Construction** (concrete composite service that ready for execution)	**Semantic enrichment of workflow** (support binding)	**Workflow** (execution of service composition definition)	**Dispatch and configuration** (organising the workflow)
					Composition testing (checking and validating the service composition)
		Execution (executing the composite service)	**Run-time** (execution engine)		**Application of composed service** (execution)
	Monitoring & exception handling		**Service management** (controlling all previous steps)		
				Mediation (handling service failure and QoS violation)	

Table 5.1: Models of the Web Service Composition life-cycles

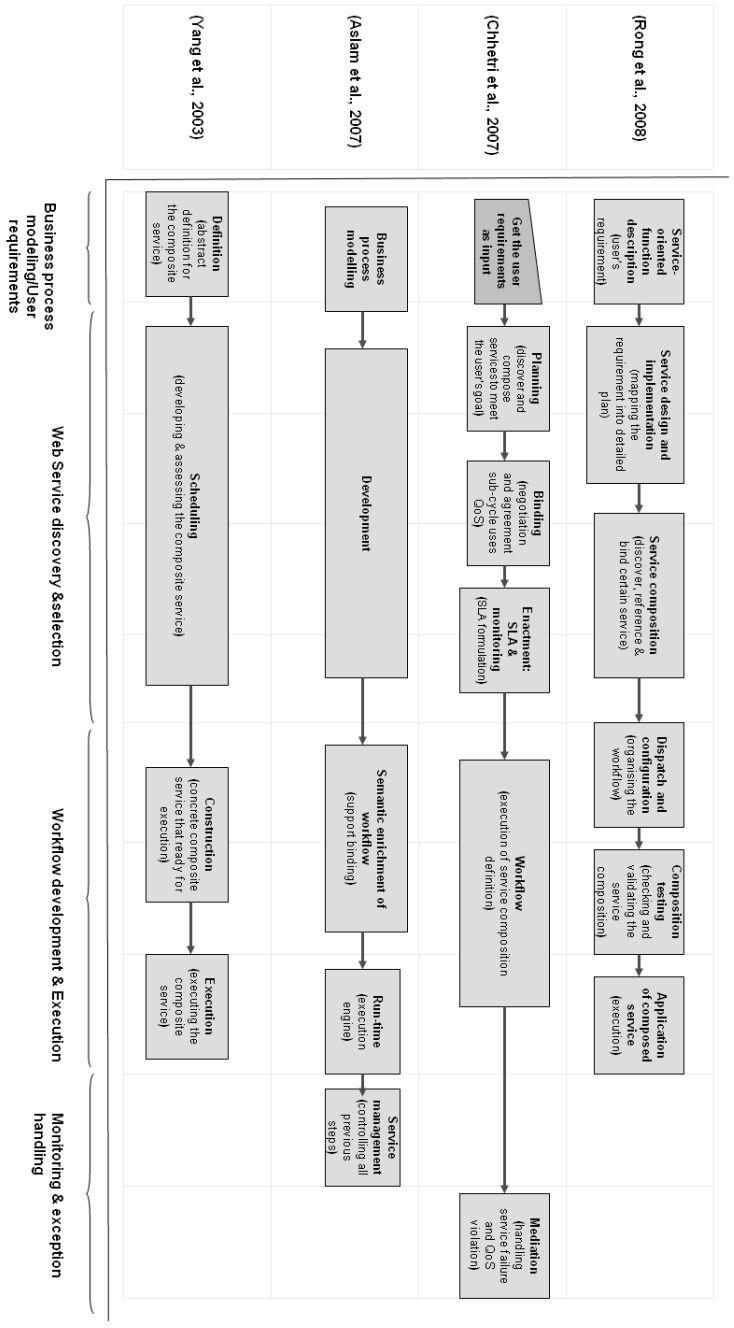


Figure 5.1: Models of the Web Service Composition life-cycle

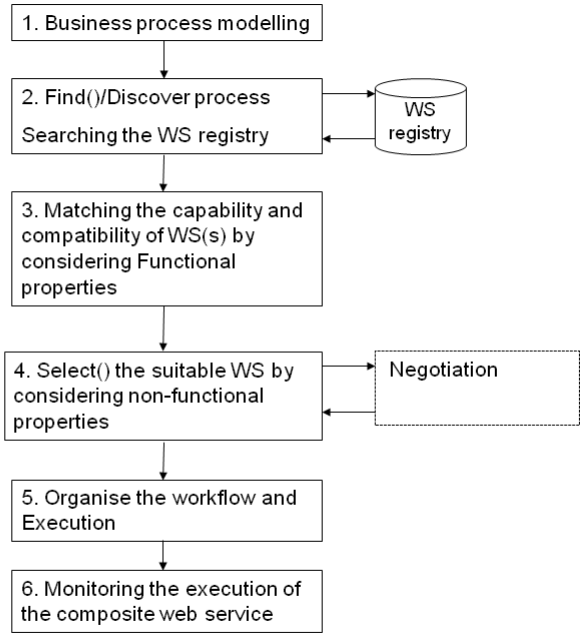


Figure 5.2: Basic steps of the Generic Composition process for Web Services

life-cycle models (see Table 5.1 and Figure 5.1 for more details). These activities are:

- Business process modelling (user requirement) step;
- Find/discovery and select step;
- Workflow development and Execution step;
- Monitoring and Exception handling step;

2. Secondly, to make these activities more granular,we re-grouped these activities into six particular groups. Each of these groups is responsible for carrying out one specific activity of the life-cycle involved in Web service composition.

3. Finally, we aggregated these activities/steps and put them in sequential order to produce a generic life-cycle model for Web service composition (see figure 5.2).

Activities/steps	Yang et al. (2003)	Aslam et al. (2007)	Chhetri et al. (2007)	Rong et al. (2008)
1- Business process modelling	✓	✓		✓
2- Find/Discovery	✓	✓	✓	✓
3- Matching	✓	✓	✓	✓
4- Select and Negotiation	✓	✓	✓*	✓
5- Organising the Workflow	✓	✓	✓	✓
6- Monitoring & Exception handling		✓	✓	

Table 5.2: The generic life-cycle model vs the four life-cycle models for Web service composition. * Has a negotiation mechanism

Table 5.2 shows how each of those four life-cycle models fits against the generic model. In the following we describe the steps of the generic model.

1. *Business process modelling (User requirement definition) step*: For this step, a Service requester/Business process developer defines his/her intended service (composite service) using an abstract description form. The description defines how services are linked with each other, as well as the organisation of control and data flow between those services needed to produce the new service. Business process modelling methods such as event driven process chain, value chain diagrams and UML are suitable ways to perform this step. However, they somehow need to be converted into an XML-based machine readable format (Aslam et al., 2007).

2. *Find/discovery step*: For this, the existing WS find/discovery mechanisms provide potential users with access to service registries that store information about businesses, services and other details (Garofalakis et al., 2004). In order to meet a composite Web service specification, the service composer searches the registries for those WSs that can achieve a given task. The enquiry that is sent to the registry consists of name of the Web Service (a registered service name in the service registry) and other parameter(s) that are used to narrow the set of discovered WSs (the outcome

from the enquiry) to a specific provider or to a specific category/domain (Wu and Guo, 2011). The outcomes should provide information about the service providers themselves, in addition to their advertised services.

3. *Matching the discovered WSs step*: This step mainly focuses on the functional properties of the discovered Web services. The outcomes from the find/discovery step contains particular information about discovered WSs,such as operation names, operation input(s), operation output(s), etc. The service composer needs to perform further evaluations to check how well the discovered WSs match the specification of an intended task (Dustdar and Schreiner, 2005). For each task in the composite Web service, the degree of similarity between the task specification and discovered Web service operation descriptions would be assessed. The traditional way of doing this is based on keyword matching, for which the service composer can use techniques, such as TF (*term frequency*) and IDF (*inverse document frequency*), to measure the degree of similarity (Hao and Zhang, 2007). After that the Web services that have the highest degree of similarity with the candidate Web services is chosen and ranked.
   The approach that has been proposed is based on the use of semantic matching for Web services discovery. This approach employs a reasoning mechanism which mainly depends on an inference engine and the integrity of the reasoning rules (Garofalakis et al., 2004, Medjahed, 2004, You et al., 2009). Another task of this step is to check the Web service capability and the compatibility of its input(s) and output(s) parameters with the rest of the business process. For instance, checking if the WS input(s) match the parameters that is used to invoke this WS in terms of context and types, and the same for the WS output(s) (Medjahed et al., 2003).

4. *Selection step*: This step is mainly concerned with using the non-functional properties and preconditions of the Web service to select an appropriate WS. Generally, the outcome of the previous step (3) results in a pool of candidate WSs contain a high degree of similarity for a specific task. Thereafter, according to the minimal similarity degree that is acceptable to the service consumer,the selection step excludes all those WSs that are failed to meet this condition. The service composer performs this step if

there is more than one WS that can achieve a specific task. Nontheless, it is worth mentiong that it may then offer different services at different costs, quality levels, etc. The select process involves performing a comparison between the remaining WS candidates and re-rank them based upon the user's preferences regarding the non-functional properties of the WS, such as response time, cost, availability, etc. (Zeng et al., 2003, Liu et al., 2004b).

The negotiation step is a complementary option before the final selection of a specific WS takes place. This step effectively extends the selection step, since both steps are concerned with the non-functional attributes of the WS. Negotiation is used to come to an agreement between the consumer and provider of the service Therefore, it helps in selecting a suitable WS according to the user's constraints and preferences. This creates a joint agreement (Web Service Level Agreement) that describes the agreed level of performance of the WS, and informs the user the appropriate actions that need to be taken if a violation of this agreement has been detected (Dan et al., 2003, Xiaogang and Tiejun, 2010).

5. *Organising the Workflow of the composite Web service step*: In this step the composed services are organised and their logical execution order is specified according to the Web service composition plan. The composition plan takes the form of Workflow that involves a combination of data and control flow which is used to manage the interaction between the composer and the selected WSs. The *bind()* operation is an essential part of the composite Web service Workflow, since this defines the message formats and protocols used to interact with the selected WS components (Medjahed, 2004).

6. *Monitoring* and *Exception handling step*: Since the WS model is based on an open environment (Singh and Huhns, 2005) - a chosen WS might not be able to deliver after being bound because of unforeseen reasons - a good composite Web service solution should be able to handle this situation (Liu et al., 2010). This step includes some wider concepts in terms of managing the execution of the composite Web service, involving exception handling, failure recovery, undo, roll back, etc. For example, if

a WS involved in a composition plan becomes unavailable or fails during execution, this step specifies what action should be taken,such as finding another WS to replace the unavailable or failed one, stopping the execution of the composition processes, etc. (Lu et al., 2007).

In the outcomes from the SLR there were 29 empirical papers that provided formal evaluation of dynamic Web service composition. Table 5.4 describes the extent to which the steps of the generic model for service composition have been covered through the approaches described in each of the 29 empirical papers. Table 5.4 shows how the service composition strategies used in each paper map on to the steps of the generic model. The first column in the table identifies the service composition strategy, the second column in the table is an index number whereas the third column in the table identifies the papers reference. The 4_{th} to 9_{th} columns identify the correlation with the six generic steps that have been identified for Web service composition. The last column is the number of steps that have been covered per paper, in addition to a symbol for each study such as one of +, +/- or - to indicate that it would produce positive results from using the technique, inconclusive ones or negative ones. These indicators (symbols) are used to represent the success of their ideas (see the tables that summarise the outcomes from the evaluations provided in each empirical paper in the previous chapter).

5.3 Analysis and Summary of the Empirical Papers

As shown in Table 5.4, the processes described in the 29 empirical papers cover all the steps that have been identified for the generic model, although it has be mentioned that no paper addresses all of them. This is largely because these papers have adopted different strategies for dynamic Web service composition. Although these papers have collectively addressed the steps from the generic model, there are two steps from this that have received less attention: *business modelling (User requirement)* and *monitoring & exception handling*. Table 5.5 shows how each step of the generic model has been covered by the six service

Strategy	No	Empirical paper — WSC Steps	1. Business process Modelling	2. Find/Discovery	3. Matching	4. Selection & Negotiation	5. Workflow & Execution	6. Monitoring & Exception handling	Total number of steps covered per paper	
Semantic Web & Ontology	1	Cheung et al. (2004)			✓	✓			2	+
	2	Medjahed et al. (2003)	✓		✓	✓	✓		4	+
	3	Ponnekanti and Fox (2004)			✓				1	+/-
	4	Segev and Toch (2009)			✓				1	+/-
	5	Williams et al. (2003)		✓	✓				2	+/-
	6	Younas et al. (2006)		✓		✓	✓		3	+
	7	Younas et al. (2005)			✓				1	-
	8	Mokhtar et al. (2007)			✓	✓	✓		3	+/-
QoS	9	Chen et al. (2006)		✓		✓	✓		3	+
	10	Gu et al. (2004)		✓		✓	✓	✓	4	+
	11	Gu et al. (2003)				✓*	✓	✓	3	+
	12	Jun et al. (2007)				✓*			1	+/-
	13	Liu et al. (2004b)				✓			1	+/-
	14	Sun (2004)				✓			1	-
	15	Zeng et al. (2003)				✓			1	+/-
Knowledge-based & AI	16	Xiaogao and Xiaopeng (2006)		✓		✓	✓		3	-
	17	Jihie et al. (2004)					✓		1	+/-
	18	Madhusudan and Uttamsingh (2006)					✓	✓	2	+
Workflow	19	Chafle et al. (2004)					✓	✓	2	+
	20	Ambite and Weathers (2005)	✓				✓		2	+/-
	21	Chi and Lee (2008)	✓				✓		2	+
	22	Fu et al. (2006)	✓				✓		2	+/-
	23	Nanda et al. (2004)					✓		1	+/-
Middleware	24	Chakraborty et al. (2004)		✓			✓	✓	3	+/-
	25	Pourreza and Graham (2006)		✓	✓		✓		3	+
	26	Williams et al. (2005)		✓	✓				2	+/-
Software Agent	27	Zakaria et al. (2004)		✓		✓	✓		3	+/-
	28	Blake and Gomaa (2005)	✓	✓			✓		3	+/-
	29	Jayaputera et al. (2007)					✓	✓	2	+/-

Table 5.4: The analysis of dynamic Web service composition empirical papers
* Has proposed a negotiation mechanism.

composition strategies' empirical papers. In the following sections we provide more details about how each step from the generic model is addressed in different papers.

5.3.1 Business Modelling (User requirement) Step

Only five out of the 29 empirical papers discuss this step. It was quite surprising as we expected that a good description of the Business Process (user requirement) would aid with providing soundness for the whole composition process in one hand and reduce the overhead for the next steps, on the other.

1. Two studies used ontologies for Web service description.
    - Ambite and Weathers (2005) used the OWL ontology language to describe the input and output of advertised Web service that address application domain-knowledge.
    - Medjahed et al. (2003) used the DAML+OIL ontology language for Web service description, also they defined an XML-based language, called CSSL for specification of a composite service.
2. Other studies used a visual formalism to capture the business process specification from the user requirement.
    - Two of these studies used Petri Nets[1] to define a conceptual service model and to create behavioural models of its components (Chi and Lee, 2008, Fu et al., 2006),
    - One study used UML class diagrams to describe the business process specification (Blake and Gomaa, 2005).

The visual formalisms adopted in these studies can easily be used by a user without expert knowledge on service composition. However, these models need an additional layer to translate the visual components into a machine readable form in order to integrate with the rest of the composition process.

[1]Petri Nets models have been a useful formalism in the information technology industry, as they are capable of presenting complex models of system processes.

Steps	Strategy	Empirical support (paper reference)
1. Business modelling	Semantic Web & ontology	(Medjahed et al., 2003)
	Workflow	(Ambite and Weathers, 2005, Chi and Lee, 2008, Fu et al., 2006)
	Software agent	(Blake and Gomaa, 2005)
2. Find/Discovery	Semantic Web & ontology	(Williams et al., 2003, Younas et al., 2006)
	QoS	(Chen et al., 2006, Gu et al., 2004)
	Knowledge-based & AI	(Xiaogao and Xiaopeng, 2006)
	Middleware	(Chakraborty et al., 2004, Pourreza and Graham, 2006, Williams et al., 2005)
	Software agent	(Zakaria et al., 2004, Blake and Gomaa, 2005)
3. Matching	Semantic Web & ontology	(Cheung et al., 2004, Medjahed et al., 2003, Ponnekanti and Fox, 2004, Segev and Toch, 2009, Williams et al., 2003, Younas et al., 2005, Mokhtar et al., 2007)
	Middleware	(Pourreza and Graham, 2006, Williams et al., 2005)
4. Select & Negotiation	Semantic Web & ontology	(Cheung et al., 2004, Medjahed et al., 2003, Younas et al., 2006, Mokhtar et al., 2007)
	QoS	(Xinjun et al., 2006, Gu et al., 2004, 2003, Jun et al., 2007, Liu et al., 2004b, Sun, 2004, Zeng et al., 2003)
	Knowledge-based & AI	(Xiaogao and Xiaopeng, 2006)
	Software agent	(Zakaria et al., 2004)
5. Workflow	Semantic Web & ontology	(Medjahed et al., 2003, Younas et al., 2006, Mokhtar et al., 2007)
	QoS	(Xinjun et al., 2006, Gu et al., 2004, 2003)
	Knowledge-based & AI	(Xiaogao and Xiaopeng, 2006, Jihie et al., 2004, Madhusudan and Uttamsingh, 2006)
	Workflow	(Chafle et al., 2004, Ambite and Weathers, 2005, Chi and Lee, 2008, Fu et al., 2006, Nanda et al., 2004)
	Middleware	(Chakraborty et al., 2004, Pourreza and Graham, 2006)
	Software agent	(Zakaria et al., 2004, Blake and Gomaa, 2005, Jayaputera et al., 2007)
6. Monitoring and Exception handling	QoS	(Gu et al., 2004, 2003)
	Knowledge-based & AI	(Madhusudan and Uttamsingh, 2006)
	Workflow	(Chafle et al., 2004)
	Middleware	(Chakraborty et al., 2004)
	Software agent	(Jayaputera et al., 2007)

Table 5.5: Summary of generic life-cycle model vs. service composition strategies

5.3.2 Find/Discovery Step

Ten out of the 29 empirical papers are included this step. Various techniques have been proposed to perform, find and discovery operations.

- Some of these techniques have adopted UDDI searching mechanism, such as the *find_service* and *get_serviceDetail* methods to find specific WS (Blake and Gomaa, 2005, Younas et al., 2006). Younas *et al.* employ a protocol that is based on a P2P architecture[2]. P2P is a form that is widely used for various Web-based distributed applications. P2P has many advantages, including dynamic communication and enhanced reliability, without suffering from single point failures, and load sharing among peer systems.

- Two studies have used a Middleware strategy such as (Chakraborty et al., 2004, Pourreza and Graham, 2006) to support Web service composition. One study that used a QoS strategy (Gu et al., 2004) was adopted a P2P architecture. These studies used the concept of P2P caching of advertisements of services and group-based selective forwarding of requests. DAML-based semantic information has been used by these studies for the selective forwarding, which is employed in the service request and service description.

- Other studies have used agent-software to deal with service discovery. It is based on agentification of the Web service environment and on semantic Web ontology (Williams et al., 2003, Zakaria et al., 2004, Williams et al., 2005). In this environment each of Web service is associated with an agent and user domain to identify which service agent can provide a specific service.

There also are two individual studies that have proposed quite different techniques for Web service discovery.

- Xiaogao and Xiaopeng (2006) used a semantic service description;

[2]P2P is a distributed application architecture that partitions tasks or workloads among peers.

- And *Chen et al. (2006)* developed an algorithm based on using an inverted file mechanism and multi-parameters search to improve the discovery process.

5.3.3 Service Matching Step

Nine out of the 29 empirical papers have discussed this step. all of them are classified under one of two of the Web service composition strategies: 1. Semantic Web ontology (Medjahed et al., 2003, Williams et al., 2003, Cheung et al., 2004, Ponnekanti and Fox, 2004, Younas et al., 2005, Mokhtar et al., 2007, Segev and Toch, 2009) and; 2. Middleware (Williams et al., 2005, Pourreza and Graham, 2006). The first group of studies have emphasised the importance of using an ontology languages for providing semantically-enriched specification of advertised Web services as the fundamental means for dynamic service matching and appropriate resource employment. These studies have also adopted a range of mechanisms, e.g. extension of ontology language, software agent and algorithms, to deal with matching step. These mechanisms focus on an input-output parameters matchmaking process between the advertised service specification and the requested service. The matchmaking process is achieved through syntactic and semantic compatibility for service input and output where:

- Syntactic compatibility in terms of data-type and correct format for data flow, i.e. where an output from one service forms an input to another service, between Web services which participate in a service composition;

- Semantic compatibility measures the degree of match (similarity) between the concepts of the advertised service and requested service. These concepts are described through the service provider domain (service domain) and the user domain by using semantic Web ontology,e.g. OWL, OWL-S and DAML+OIL. Therefore, the matchmaking process takes place using subsumption relationships presented by the ontologies (service and user domain) in order to match the advertised services with the user request.

5.3.4 Select and Negotiation Step

Thirteen out of the 29 empirical papers addressed this step. Table 5.4 shows that the empirical papers that used a QoS strategy for Web service composition are the main contributors for this step.

1. All of the seven papers that used a QoS strategy address this step. Various techniques have been proposed by these papers to compute the attributes of the service (non-functional properties) that are considered while selecting the service, most appropriate for the user preferences. These approaches are:

    - Zeng et al. (2003), Liu et al. (2004b) have proposed a *QoS-model* that includes certain attributes/qualities, such as execution price, execution duration and reputation, as a part of the Web services specification. By using calculation formulas for each similar Web services that provide same functionalities for a specific task, these qualities are calculated to differentiate these similar Web services according to the values of their qualities.
    - Other approaches make the assumption that the QoS data is provided and assured by service providers, which is computed based on execution monitoring by user/third party or aggregated through requesters feedback. It depend on the characteristics of each QoS attribute (Gu et al., 2004, Sun, 2004, Chen et al., 2006). Again similar to the studies mentioned above, they employ some mechanism, e.g. algorithm, protocol and software-agent, to calculate the QoS attributes for selecting suitable Web service.
    - Only two of the QoS strategy papers have addressed the Service Level Agreement (SLA) (Gu et al., 2003, Jun et al., 2007). Based on their SLA contracts, an appropriate service is selected to take part in service composition.

2. Different approaches for the Selection step have been proposed in those studies that use other service composition strategies:

- Some of the empirical papers that use a semantic Web ontology strategy for service composition also proposed mechanisms for service selection (Medjahed et al., 2003, Cheung et al., 2004, Younas et al., 2006, Mokhtar et al., 2007). These mechanisms are based on semantic match of the service capabilities with the user task. A services selection function based on QoS specification, selects the most appropriate service among those services that match the user request.
- Another study used a knowledge domain approach (Knowledge-based and AI strategy) to pilot the service composition process and give advice on service selection (Xiaogao and Xiaopeng, 2006).
- Lastly, one study based on agentification of the service composition environment (Software agent strategy) described in (Zakaria et al., 2004) used a software agent (user agent) to interact with service providers (provider agents) selecting the most appropriate service based on a single quality attribute (cost).

5.3.5 Organising the Workflow and Execution Step

A large proportion of the 29 empirical papers (19 papers) have addressed this step . These papers have proposed a range of solutions, targeting the different natures of the service composition environment. This includes, Grid computing network, home area network/computer lab and peer-to-peer network architecture.

1. Three of the papers that used a semantic Web ontology strategy have addressed this step (Medjahed et al., 2003, Younas et al., 2006, Mokhtar et al., 2007). The strength of the Workflow and execution solutions that they have been employed for service composition are based on a semantic ontology to enhance the processes of finding and selecting appropriate services to be involved in a service composition. Their solutions worked by semantically enriching the service description and the control flow specification of the composite services.

- Medjahed et al. (2003) extended the WSDL language to support the specification of Workflow.
- Younas et al. (2005) proposed a protocol to support the composition process in a Peer-to-Peer (P2P) network architecture.
- Mokhtar et al. (2007) proposed a mechanism to deal with the diversity of services in pervasive computing environment, such as printer, PDA device[3], personal computer etc., to support services integration.

2. Three of the papers that used a QoS strategy addressed this step (Gu et al., 2003, 2004, Chen et al., 2006). Their techniques were based on identifying a set of high quality services that need to be composed in order to achieve a requested service.

   - Gu et al. (2003, 2004) proposed two different protocols (centralised and decentralised, respectively) to deal with this step based on a P2P network architecture. These two protocols are quite similar in terms of identifying the qualified path (composed services plan) over the P2P network, based on functionality and QoS (select step). The centralised protocol uses a service composer to orchestrate the performance of the composite service. The decentralised protocol, on the other hand, initialises service components at each intermediate peer, and then streams application data units along the selected path (execution moves from one node to another until it is executed).
   - Chen et al. (2006) used an inverted file data structure to build a Web Service Composition Tree (WSCT) and then used an algorithm to get the best Web Service Composition Result (WSCR) in terms of QoS based on user request.

3. To some extent the studies that used a knowledge-based and AI strategy employed similar approaches to deal with this step (Jihie et al., 2004, Madhusudan and Uttamsingh, 2006, Xiaogao and Xiaopeng, 2006). These

[3]Personal Digital Assistant device

approaches combined a knowledge-based approach (employ domain knowledge for developing enough of a shared vocabulary) and AI planning techniques (tracking relations and constraints among individual step) to facilitate dynamic and scalable Web service composition. They used a semantic ontology assisted by knowledge-based (rich presentation of service component) to find the appropriate service components. Thereafter, they used an AI planning technique to search for the best service composition plan. Similarly, they used Workflow engine to construct the appropriate Workflow.

4. The studies that used a Workflow and composition language strategy proposed several different approaches to deal with this step.

   Three studies proposed a centralised approach to organise and execute the Web service composition (Ambite and Weathers, 2005, Fu et al., 2006, Chi and Lee, 2008).

   - Ambite and Weathers (2005) relied on the BPEL4WS standard and IBM BPWS4J engine to describe and execute the Web service composition step.
   - Fu et al. (2006) and Chi and Lee (2008) used similar techniques which were based on the use of Petri Nets to model and validate the description of the Workflow of the composed service., By using a given mechanism (a translator tool such as PNML[4] and Maude[5]), they translated the Workflow into a machine-executable format (BPEL).

   Two studies proposed a decentralised approach to organise and execute the Web service composition step (Chafle et al., 2004, Nanda et al., 2004). They used BPEL4WS to describe the Workflow of the service composition, The BPEL4WS code is divided into a number of partitions. Partition for each logically related sequence of processes in the code. Then, these partitions are executed by using same number of BPWS4J engines that are allocated on certain computational nodes. These nodes take part on

[4]http://www.pnml.org/
[5]http://maude.cs.uiuc.edu/

the service composition which are responsible in performing a particular part of the composition.

5. Two studies that used middleware strategy proposed a centralised approach (Pourreza and Graham, 2006) and decentralised approach (Chakraborty et al., 2004) to deal with this step.

   - The centralised approach uses a middleware for local interaction environments, such as a Home Area Network (Pourreza and Graham, 2006). The middleware is embedded into a Home gateway device (HGD[6]) which is responsible of description and deployment of the service composition. The description is sent over the internet to a third party (service provider) to generate the service composition Workflow, and then the same description of the Workflow is sent back to the middleware. The middleware has an execution engine that executes and monitors the execution of composite services in the home.
   - The decentralised approach employs a Broker-based protocol for pervasive environment (ad-hoc network) (Chakraborty et al., 2004). A single node is elected to work as a broker to organise and coordinate with the composition of the services discovered from the previous step. And then the execution of the individual services happens in a distributed manner at the nodes hosting those services.

6. Three studies using a software agent strategy proposed similar approaches to deal with this step. The proposed solutions were based on the agentification of the Web service composition (Zakaria et al., 2004, Blake and Gomaa, 2005, Jayaputera et al., 2007). The client-agent is responsible for coordinating and controlling the Workflow of the composite service. After all services (each service presented by service-agent) that are involved in the composition have been identified. The client-agent is responsible of executing the composition Workflow by binding and interacting with services-agents.

[6]A home gateway device which is roughly analogous to a wireless access point though it may offer storage and other services as well as Internet connectivity.

5.3.6 Monitoring and Exception Handling Step

Only six out of the 29 empirical papers implemented the monitoring and exception handling step. They employed a range of failure recovery mechanisms, reflecting the different service composition environments.

- Gu et al. (2003) implemented a dynamic service composition approach for an overlay network[7] that has the ability to recover from failure when a service outage or a QoS violation happens by directing the execution to another node.
- Gu et al. (2004) proposed a proactive failure recovery mechanism to maintain the quality of composed service during run-time. The mechanism keeps a small number of service backups for each task. Accordingly, for failure recovery, the execution can be quickly redirected to another service (from the backup) rather than using the broken service.
- Chafle et al. (2004) recommended a central entity that monitors the execution status of the composed service. When an error occurs, then this entity stops the execution and invokes the 'undo' routine to remove any changes that would have happened.
- Chakraborty et al. (2004) used a Broker to monitor execution of the service composition. When an error occurs, then the Broker commits any completed part and re-executes the uncompleted part.
- Madhusudan and Uttamsingh (2006), Jayaputera et al. (2007) suggested similar techniques. These techniques deal with a error when it occurs by modifying the plan and re-execute the whole composed service.

5.4 Summary

We have proposed a generic life-cycle model for service composition which includes all the activities/steps essential to performing a Web service composition. This model has been validated by comparing it with the four life-cycle

[7] An overlay network is a computer network which is built on the top of another network.

models that have been identified through the review of the related literature . Regarding the process of service composition, the activities of the generic life-cycle model are more comprehensive than the activities/steps of these life-cycle models. The generic life-cycle model and its steps have then been used to analyse with more detail, showing that how extensively the 29 empirical papers have covered the activities/steps of service composition.

An analysis of the 29 empirical papers shows that these papers proposed various mechanisms for dynamic service composition due to the fact that they studied different problems. In addition, the service composition environments that have been studied are quite different. such as Grid computing network, Home area network/Computer lab, and Peer-to-Peer network architecture which required to be approached differently. The literature shows that three service composition strategies, semantic Web and ontology languages, QoS, and Workflow and composition languages, are the major contributors for the Web service composition in terms of number and depth the service composition have been studied. Furthermore, these strategies showed great impact on the correctness and quality of the composite service. In particular:

- The semantic Web and ontology languages empirical papers showed great impact on the find/discover and match processes which in turn demonstrated potential for composing numerous Web services and Web contents to generate a more complex system. Thus, the semantic description of the Web service should be fully recognised and adopted by the Web service communities.

- The quality-based selection of Web service components is an important issue in dynamic Web service composition. The QoS empirical papers showed potential improvements on the quality of the composite service. However, there is still no common agreement about which qualities (non-functional properties) should be advertised and used with the Web services description, and how they should be computed. In following chapters we will discuss in details the impact of using a QoS-model to improve the quality of composite service.

- The Workflow and composition languages empirical papers mainly fo-

cused on business process modelling and the organisation of the Workflow. Their solutions for business modelling process were aimed to provide robust service composition without the need for significant design time or skill of developers. In addition, they employed validation mechanisms with their solutions to enhance the reliability of Web service composition.

- Although, the rest of the composition strategies propose promising solutions for Web service composition, they are either expensive (Middleware strategy) or require great effort and expert developers to build them (the examples are, Knowledge-based and AI and Software agent strategies).

Chapter 6

Evaluation of the Empirical Papers For A Replication Study

6.1 Introduction

This chapter investigates the uses of simulation as a research method for software engineering. Furthermore it provides a background about replication study and its feasibility for creating and extending scientific knowledge. The chapter examines how researchers have used simulation experiments to validate their ideas. In order to choose a paper for conducting a replication study through the use of simulation, an analysis of the papers from two service composition strategies is presented.

6.2 Using Simulation as a Research Method

In this section, we provide a background of simulation and show that how it is interpreted in software engineering. A brief summary of some examples from empirical papers that used simulation to evaluate their studies is also provided.

6.2.1 Simulation (Background)

Lazić and Mastorakis (2005), and Reardon et al. (2010) suggested that there are many reasons for using simulation to study a system. Such as, when a system is very complex and not easy to understand, or when the application or platform is not easily obtainable by the experimenter. Other reasons for using the simulation would be due to cost or time constraints. Reardon *et al.* also emphasised the importance of using simulation in support of software engineering. There are many aspects of software engineering where simulation can play a significant role. This includes the "requirements specification, process improvement, architecture trade-off analysis and product line practices" (Reardon et al., 2010). In addition, they acknowledge that there are now sufficiently mature commercial simulation applications that are capable of supporting software development needs with low cost, easy to use and readily available.

A study performed by Abu-Taieh and El-Sheikh (2010) indicates that, since the 1970s, simulation education has moved more into the spotlight. This is a consequence of an increasing acceptance of Modelling and Simulation (M&S) studies across many major application domains, e.g. military, industry, services, and various scientific fields. Consequently, the demand for well-qualified specialists has increased. However, Molnar observes that "the place and recognition of M&S, however, is not very well recognized by academics; M&S as scientific disciplines are 'homeless' " (Molnar, 2010). This reveals and emphasizes the multidisciplinary and interdisciplinary character of M&S, which can make this approach less attractive for educational programmes and curriculum development.

Simulations have been used both for off-line analysis to study systems design, as well as to explain the effects of change and the redesign of operating policies. Where reliable statistical analysis tools are available, simulation technology may be beneficial to use for developing real-time embedded control systems for large-scale applications. For such cases, metrics of performance depend on data obtained in both temporary and steady-state periods (via simulation). It may be used for the development of real-time control policies to deal with a large-scale system effectively (Averill M Law 2008).

6.2.2 Some examples of Simulation Empirical Papers

To help identify suitable models for a simulation experiment, we have conducted a brief survey by examining a set of issues of the Journal of Systems & Software to search for papers that employed simulation for software engineering. This is a journal that publishes many papers where the simulation is used as their research method. We limited our search to the period from January 2011 to April 2012. We found 149 papers that used simulation. Out of which 46 papers were selected, based on their provision of a specific section that described their simulation experiment. After re-examining them, we reduced our selection of papers that developed a simulator rather than using an existing simulation package, along with providing a description of the simulation set-up and/or data-sets used. This selection criterion, therefore, reduced the number of papers to 11, which were finally selected..

The aim of the analysis of the simulation papers was to study how simulation has been used in software engineering. This helps in devising a simulation framework that can be used to study Web service composition. The analysis is based on a number of guidelines for conducting experiments in general, which are further discussed in Section 6.5, and on the simulation experiment design guideline that is proposed by Barton (2010). The following are the elements of the analysis:

1. Purpose/Goal for using simulation;
2. Simulation model formula;
3. Input list that is used in the simulation;
4. The type of data-set that is used;
5. The measurements (the outputs);
6. The presentation of results

We analysed the outcomes of the survey of simulation papers using these factors. Table 6.1 summaries 11 simulation papers in more detail. The analysis describes how each of these paper complied with the simulation elements, and the aspects

of software engineering that have been covered (procedures, measurements and presentation of the results).

Table 6.1: Analysis of simulation empirical papers design

Paper	**Purpose/Goal**	**Simulation model formula**	**Input list**	**Data-sets**	**Measurements**	**Result presentation**
Lin and Tang (2011)	The effectiveness of their approach in a hybrid sensor network	polynomial-time algorithms	✔	-	Coverage ratio, Movement cost and Energy consumption	- Line chart
Chen and Weng (2012)	Reduce power consumption in wireless multi-hop networks	Linear algorithm	✔	-	Throughput, Power consumption and Network lifetime.	- Line chart
Wang et al. (2012)	Investigate adaptive model-free approaches for resource allocation and energy management in multi-tier cloud environments	Sigma function of divisors	✔	Online data	Resource capacity allocation, Total revenue, Status of physical nodes and Admission control effect	- Linear chart - Table of some outputs
Moadeli and Vanderbauwhede (2011)	An analytical model to predict the average latency of wormhole-routed in Network on chip (NoC)	Sigma function of divisors	-	Exponential distribution	Average broadcast latency	- Linear chart
Park and Bae (2011)	Identify the affected elements of a software process using process slicing	Sigma function of sum	-	Real project data	Cost, Duration, and the Number of Defects	- Table of outputs

Continue in next page...

Paper	**Purpose/Goal for using simulation**	**Simulation model formula**	**Input list**	**Data-sets**	**Measurements**	**Result presentation**
Zhu et al. (2011)	Fault-tolerance scheduling to improve system resource utilization and schedulability	Linear algorithm	✔	-	Guarantee ratio and Overall performance impact of node number	- Linear chart
Ooi et al. (2012)	Enhancing service availability and reducing low-level decisions making problems from administrator.	Sigma function of divisors	-	Experimental data	Availability, Cost and Performance	- Linear chart - Histogram - Table of some outputs
Liu and Zhang (2012)	Enhancing the performance of the k nearest neighbour (k-NN) rule for classification mining	Heterogeneous Euclidean-Overlap Metric function (HEOM)	-	benchmark data-sets	Accuracy of Noisy data elimination	- Histogram - Table of some outputs
Du et al. (2011)	Optimized QoS-aware replica placement heuristics and applications in astronomy data grid	Sigma function of sum	✔	Experimental data	Replication cost and Execution time	- Linear chart - Table of some outputs
He et al. (2012)	Improve schedulability in energy constrained distributed real-time embedded systems	Sigma function of divisors	✔	Simulated data	Guarantee Ratio, Energy consumption and CPU Utilization	- Linear chart - Histogram
Huang et al. (2011)	Analysis of middleware service impact on system reliability	Sigma function of sum	-	Experimental data	Service failure and Response time	- Histogram

Last part of Table 6.1.

Table 6.1 shows that simulation have been used across many topics in Software

engineer. The simulation usage varied from enhancement, improvement and evaluation of the performance of existing systems or to predict the behaviour of under-development systems. All studies used mathematical formula to develop theirs simulation model. Although the listing of the inputs (independent variables) and their values help the readers to understand how the experiment has been controlled, however, the table shows that six out of eleven papers did so. The presented data-sets have two forms; real data, which come from actual observations (real project data, on-line and benchmark data-sets); and data-sets that are generated by algorithms, which are used in that case, where it is hard to obtain in terms of availability of in new type of formats. The presentation of the results has three ways; linear chart, histogram and table of outputs. We found that the linear chart was the most popular presentation that has been used here.

6.3 What is a Replication Study?

A *Replication Study* is the repetition of an empirical study. in the majority of cases the replication study is carried out within a different situation and with different participants or data-sets, aiming to check the generality of the finding from the original study. Numerous studies discussed the role of the replication study in empirical software engineering (Brooks et al., 1994, Shull et al., 2002, Miller, 2005, Mäntylä et al., 2010). Shull et al. and Mäntylä et al. both mentioned that the published literature on replication studies has been increasing in recent years. Consequently, the importance of replication studies has also gained greater recognition within the empirical software engineering community.

The significance of replication has long been recognised throughout the natural, social and the engineering sciences as a method of creating and extending scientific knowledge (Mäntylä et al., 2010). Moreover, these authors noted that it was rare to find any serious scientist who does not regard replication as a fundamental ingredient of scientific work. Mäntylä et al. (2010) quote Popper's (1959) statement "*We do not take even our own observations quite seriously, or accept them as scientific observations, until we have repeated and tested them.*

Only by such repetitions can we convince ourselves that we are not dealing with a mere isolated "coincidence," but with events which, on account of their regularity and reproducibility, are in principle intersubjectively testable".

6.4 Classification of Replication Studies

The roles performed by replication studies have been classified in a number of ways. Lindsay and Ehrenberg (1993) and Mäntylä et al. (2010) have classified replication studies as being either exact/close or conceptual/differential replicated experiments (studies).

- The exact/close form aims to maintain almost all known conditions of the original study that are same or to a great extent similar in the replication. For example, the population or populations in question, the sampling procedure, the measuring techniques, the background conditions and the methods of analysis. (Lindsay and Ehrenberg, 1993). An exact/close replication can then be used to verify results of an original study.

- The conceptual/differentiated form is one where variations occurs in significant experimental variables, or it replicates only the research question of the original study. The goal is to expand the scope of conditions under which the results are still valid. Investigating the effect of intentional variations in the experimental conditions provides the basis for generalisation (Lindsay and Ehrenberg, 1993). Mäntylä et al. acknowledge that conceptual replications provide less scope for comparison when there are different results as a consequence of following different procedures.

According to Brooks et al. (1994) there are two forms of replication study, *internal and external*:

- Internal replication means it is carried out by the same researcher(s) who conducts the original study;

- External replication means it is carried out by an independent researcher(s). doing this is crucial for establishing sound results.

Shull et al. (2002) discussed how the availability of documentation of an experiment (laboratory packages) can promote better replications and complementary studies. "A laboratory package describes the experiment in specific terms and provides materials for replication, highlights opportunities for variation, and builds a context for combining results of different types of experimental treatments. Laboratory packages build an experimental infrastructure for supporting future replications" (Shull et al., 2002). However, their experience also demonstrated that such a goal for replication was ambitious, and that providing laboratory packages was not the solution in itself.

Even when both the original and replicator researchers are experienced experimentalists, it is highly likely that a number of sources of variation and unexpressed assumptions relating to the experimental context. This means that setting up a static laboratory package to outline all relevant aspects of the experiment so that unexpected sources of variation are not introduced into the replication, is almost impossible. Although, the laboratory packages are very important for supporting an exact replication process, other factors need to be considered as well. They include, involving the original researchers to support their instantiation, evolution and use. The requirement for such a process indicates that there should be some form of collaboration between the original and replicating researchers to transfer the tacit knowledge that is essential for process conformance.

Kitchenham (2008) agrees with Shull *et al.* about the importance of replication studies as a basic component of the scientific method However, at the same time he disagrees about the value of close replications, on account of the risk of reinforcing and reusing possible flaws in the original experimental design which consequently would be introduced into the replications. Both Kitchenham (2008), and Juristo and Vegas (2009) noted that, due to variations in experimental conditions, nearly all experimental replications among researchers undertaken at different locations have not been satisfactory. They conclude that "identical replications are almost impossible to achieve and thus senseless as non-identical replications also can generate new knowledge". Taking into account these two different views about replication, the likely outcomes of a proposed replication study and its contribution to knowledge are important

factors in determining feasibility.

The aim of this thesis is to explore the use of simulation to study the Web service composition. Activities aim involves two steps:

- Performing a close replication by simulation in order to give confidence that the simulation process produces results that are consistent with those from the original application that helps to validate the simulation process.
- Perform one or more differentiated replications to explore the effect of changing the values of key features, where the simulation is used to help scope out the effects of changes in the Web service composition process itself.

6.5 Selection of a Study for Replication

In describing the outcomes of the SLR, Table (4.3) that presents the analysis of dynamic Web service composition empirical papers—shows that the Semantic Web and Ontology and the QoS strategies are the major forms used in service composition research, at least in terms of the number of studies. In addition, they have the larger proportions of empirical papers that produced positive results by using these techniques for Web service composition. Therefore, they are the most useful candidates for a replication study. In this chapter, we examine the empirical papers from these two strategies to choose one paper for replication study, and to identify how they could be extended for further improvement. Therefore, we are looking for a study that provides adequate information to allow the possibility for a close replication as well as having scope for extension. As we have mentioned earlier, this information needs to be available through the documentation of the experiment (in the form of laboratory packages of experimental materials, experiment report, etc.) in order to replicate a study (Shull et al., 2002, Miller, 2005, Mäntylä et al., 2010). There are different sets of guideline for reporting the experiments that identified some criteria which can assist the researchers to document their experiment (Kitchenham et al., 2002, Jedlitschka and Pfahl, 2005, França and Travassos,

2012). The following is an outline of the key experiment reporting criterionwe can derive from these guidelines:

1. Goals, hypothesis and theories
2. Experimental plan (Design, Data collection and analysis)
3. Experimental procedures
4. Results

These criteria to some extent are similar to the outline of a reporting scheme for experiments that is presented by Miller (2005) (see Table 6.2). However, Miller gives more details that relate to our purpose. It matches these criteria against the empirical papers for the two composition strategies in order to choose one of them for our replication study.

1. Goals, hypothesis and theories
A. Aspects of goal
B. Hypotheses
C. Theories
2. Experimental plan
A. Experimental design
B. Treatments
C. Objects
D. Subjects
E. Data collection and validation
F. Data analysis
3. Experimental procedures
A. Training activities
B. Conducting the experiment
C. Feedback to subjects
4. Results
A. Data
B. Interpretations

Table 6.2: Overview of Characterization scheme for experiments as presented by Miller (2005)

We used a qualitative assessment to evaluate these papers in terms of reporting their experiments. . The assessment uses three values to indicate how far each paper has complied with the experiment reporting criteria described above:

- *Good*, means the paper has provided enough details.
- *Fair*, means the paper has provided some details.
- *Poor*, means the paper has provided little details.

The columns in Tables 6.3 and 6.4 list these as evaluation criteria for two groups of empirical papers. There is also another column which describes potential extensions that can be added to these empirical papers. The data in this column has been obtained by examining the discussions about future work and results provided in the conclusion section of each paper.

Tables 6.3 and 6.4 provide assessments/evaluations for the papers from the two strategies (Semantic Web & ontology and QoS papers). The evaluation is focused on the reporting criteria of their experiment in order to choose one study for a close replication. The assessment helps to identify five papers for potential replication. This exercise is undertaken considering that they provide more information about their study than the rest of the empirical papers from the same strategies. The identifications guided by the simulation experiment elements and the analysis of the simulation papers are mentioned earlier (see section 6.2.2), as welll as the outcomes reported in these tables. For identification of candidate studies, we looked at the technical information provided by each paper related to the information needed to perform a replication study. These technical elements (summarised from table 6.2) are:

- Method (the details of the approach);
- The experimental set-up;
- The experimental parameters (independent variables) and their values (data-set);
- The outputs (dependent variables).

All of these five papers implemented prototype systems in order to verify their approaches for dynamic Web service composition. Some of these papers used their prototype to experimentally evaluate their proposed solutions (Zeng et al.,

Paper	**1. Goals, hypothesis and theories**	**2. Experimental plan**	**3. Experimental procedures**	**4. Results**	**5. Potential Extension**
Cheung et al. (2004)	Good	Good	Fair	Good	- Integrate planning with matching. - Service scoring approach
Medjahed et al. (2003)	Good	Good	Fair	Good	- Include additional semantic - XML-schema's user defined data-type and define data-type compatibility among message parameters
Mokhtar et al. (2007)	Fair	Good	Fair	Fair	N/A
Ponnekanti and Fox (2004)	Good	Good	Fair	Poor	N/A
Segev and Toch (2009)	Good	Fair	Fair	Good	- Further prototyping and evaluation - Create ontology for service-oriented concerns such as location and access protocol.
Williams et al. (2003)	Good	Fair	Fair	Good	N/A
Younas et al. (2005)	Good	Good	Poor	Poor	- Test the system in an open environment - Further expansion for mismatch role
Younas et al. (2006)	Good	Good	Fair	Fair	- Deployment of the composition mechanism on the pervasive computing environment

Table 6.3: Semantic Web and ontology language papers vs. Experiment reporting criteria

Paper	1. Goals, hypothesis and theories	2. Experimental plan	3. Experimental procedures	4. Results	5. Potential Extension
Chen et al. (2006)	Good	Good	Fair	Fair	- Implement fuzzy matching - Include service semantic in the matching process
Gu et al. (2003)	Good	Fair	Fair	Good	- Work to support secure service composition - Support more expressive service composition semantic
Gu et al. (2004)	Good	Fair	Fair	Good	N/A
Jun et al. (2007)	Good	Fair	Fair	Fair	- Further exportation to SLA - Integrate the approach with ASAPM[1]
Liu et al. (2004b)	Good	Good	Fair	Good	- Automate the collection of the feedback data
Sun (2004)	Good	Fair	Poor	Poor	- Build up a detailed QoS composition profile from the business aspect, architecture aspect and technology aspect
Zeng et al. (2003)	Good	Good	Fair	Good	- Exception handling

Table 6.4: QoS papers vs. Experiment reporting criteria

Paper	Method	Experiment set-up	Experiment paramet-ers	Output
Cheung et al. (2004)	Mostly	Partly	Partly	Mostly
Medjahed et al. (2003)	Mostly	Partly	Mostly	Mostly
Liu et al. (2004b)	Mostly	Partly+	Mostly	Mostly
Sun (2004)	Partly+	Partly	Little	Partly
Zeng et al. (2003)	Mostly	Partly+	Mostly	Mostly

Table 6.5: The five empirical papers vs. the experiment elements

2003, Liu et al., 2004b, Cheung et al., 2004, Sun, 2004). One study built a simulation testbed rather than using their prototype for the experiment (Medjahed et al., 2003). We observed that all these studies used simulated data-set. This is because there is no enough real-system data of Web service, or data is not available data in the form of extended Web service descriptions (semantic/ontology descriptions) that can be used to perform their experiment.

Table 6.5 shows how well each paper meets each of these experimental elements. The columns represent these elements with qualitative values (mostly/partly/little) to describe how well these elements are presented in each paper. The first two papers are slected under the Semantic Web and ontology strategy. The last three papers are selected under the QoS strategy. The outcome of this table also assists with producing a ranked list of these five empirical papers for potential replication. The paper that meets most these technical elements is in the top of the list.

1&2. Zeng et al. (2003) and Liu et al. (2004b)

3. Medjahed et al. (2003)

4. Cheung et al. (2004)

5. Sun (2004)

Brooks et al. (1994) and Miller (2005) claimed that experiment reports must offers enough information in order to allow other researcher to evaluate the study or perform an external replication. Brooks et al. stated that "Unfortunately, numerous empirical studies in the software engineering literature are lacking in this respect", which is similar to the outcome from these tables. (6.3, 6.4 and

6.5) showed that all these empirical papers from the two categories (including the listed papers) are not fully reported their experiments. However, after producing this list, we contacted the authors of these five papers via email for a collaboration. The collaboration helps us getting the information that we need along with the help of transferring the tacit knowledge that are not reported in the papers (Shull et al., 2002). The reply from the authors were as follows:

- No reply from (Zeng et al., 2003) authors.
- No reply from the first author of (Liu et al., 2004b). The second author however replied in saying "The set-up is running on an old version of BEA server. I am happy to host you as a student in the summer for a month to work on this problem in my lab if you can find your own financial source to come here".
- We received a positive reply from (Medjahed et al., 2003) authors, and the first author was willing to answer all the further questions.
- The first author of Cheung et al. (2004) just replied with the same information that we found in their paper. He said "we followed quite close to what being mentioned in the paper".
- We did not contact the authors of Sun (2004), as shown in table 6.5 that the reporting elements were not good enough that we could replicate the original study.

Taking into account the ranked list of the five papers and the results of contacting the authors of these five papers, we found that the reply from (Medjahed et al., 2003) authors made it most suitable paper for a replication study. The replication study eventually empirically evaluates the result of the original study and makes the comparisons between the results of the repliced and original study in order to investigate whether the outcomes of the original study can be generalised or not.

6.6 Summary

The outcomes of the analysis of the empirical papers of the Semantic Web and ontology and QoS categories, as shown in table (6.3,6.4), result in nominating five empirical paper for a close replication study. Thereafter, we evaluated these five papers according to the experiment reports of their study, as shown in table 6.5, and then put them in an ordered list. The paper with a high quality report was presented in the top of the list. In addition to the evaluation, we took into account the replies of the authors of the five papers who answered our questions about their experiments. Therefore, our final decision was to select (Medjahed et al., 2003) study for a close replication.

Chapter 7

A Simulation to Replicate a Web Service Composition Study

7.1 Introduction

The use of simulation for replication is addressed in this chapter as well as in the following chapter . This chapter addresses the use of simulation as a research method for performing a close replication study. The baseline study that is used for this is presented by Medjahed et al. (2003) "Composing Web services on the Semantic Web". In the following chapter, we describe an extension study to our replication of the aforementioned study by using QoS model to organise the selection phase of the service composition process. The simulation experiments for both the close replication and the extension studies is reported in some detail. The reporting is followed the experiment report guidelines that were mentioned in the previous chapter. It also presents the details about the context of the experiments and the problems that have been examined. It identifies the independent and dependent variables, and the measures that were used in the experiments. Also we describe the simulated data-sets that were used through the experiment. Finally we describe the experimental procedure involved, and present the results.

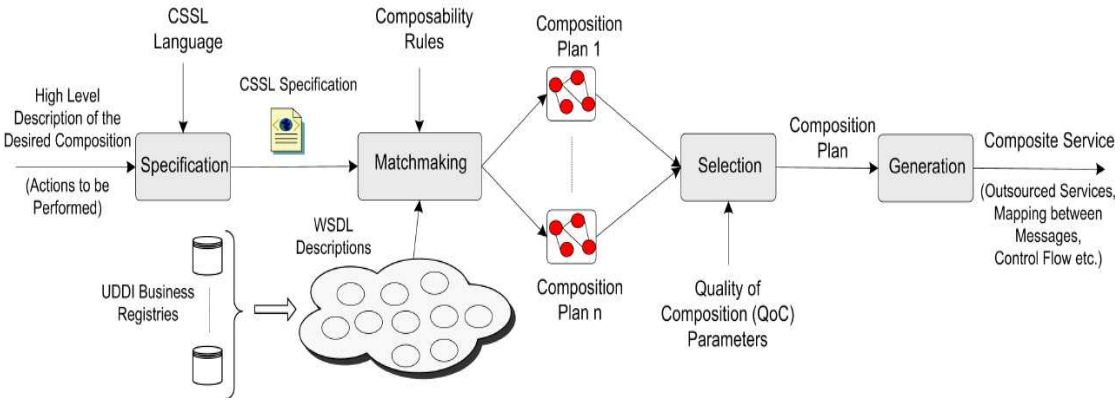


Figure 7.1: Overview of the proposed approach for service composition (Medjahed et al., 2003)

7.2 The Baseline Study

As mentioned in the earlier chapters, the WSDL language has been recommended as the standard way to describe Web services. A WSDL description consists of service name, operation, message, inputs and outputs (W3C, 2001). However, WSDL offers little or no support for providing a *semantic* description of a Web service (Akkiraju et al., 2005, Bing and Huaping, 2005, Dumez et al., 2008). It basically provides a grammar that can be used to describe a Web service from a syntactic point of view. The literature shows many approaches that have been introduced to cater for Semantic Web-enabled Web services. Most of these approaches have extended WSDL by adding semantic capabilities which provide a basis for the *automatic* selection and composition of Web services. In the paper that has been used as our baseline, Medjahed et al. proposed such a framework for the automatic composition of Web services. The core idea behind their approach is based on combining the emerging concepts of Web services and ontologies as shown in Figure 7.1.

Medjahed et al. (2003) provided a number of definitions that can be used to add a semantic description for a Web service which can then be used through the matchmaking process. Here, we restate five of these definitions, since they represent the core part of the matchmaking process.

Definition 1: *Web service.* A Web service WS_i is defined by a tuple ($Description_i$, OP_i , $Bindings_i$, $Purpose_i$, $Category_i$) where:

- $Description_i$ is a text summary about the features of the service.
- OP_i is the set of operations provided by WS_i .
- $Bindings_i$ is the set of binding protocols supported by WS_i .
- $Purpose_i = \{Purpose_{ik}(op_{ik}) \mid op_{ik} \in OP_i\}$ is the set of purposes for the WS_i operations.
- $Category_i = \{Category_{ik}(op_{ik}) \mid op_{ik} \in OP_i\} \cup \{Category_i\ (WS_i)\}$ is the set of categories for the WS_i operations.

Definition 2 – *Binding composability.* Two services $WS_i = (D_i\ ,\ O_i\ ,\ B_i\ ,\ P_i\ ,\ C_i\ )$ and $WS_j = (D_j\ ,\ O_j\ ,\ B_j\ ,\ P_j\ ,\ C_j\ )$ are binding composable if $B_i \cap B_j \neq \phi$.

Definition 3: *Operation.* An operation OP_{ik} is defined by a tuple ($Description_{ik}$, $Mode_{ik}$, In_{ik}, Out_{ik}, $Purpose_{ik}$, $Category_{ik}$, $Quality_{ik}$) where:

- $Description_{ik}$ is a text summary of the operation features.
- $Mode_{ik} \in \{$*"one-way", "notification", "solicit-response", "request-response"*$\}$. In_{ik} and Out_{ik} are the input and output messages, respectively. $In_{ik} = (\tau_{ik},\ \phi)$ and $Out_{ik} = (\phi, \tau_{ik})$ for notification and one-way operations, respectively.
- $Purpose_{ik}$ describes the business function offered by the operation.
- $Category_{ik}$ describes the operation's domain of interest.
- $Quality_{ik}$ gives the operation's qualitative properties.

Definition 4: *Message.* A message M is defined as a tuple *(P,T, U, R)* where:

- P is a set of parameter names.
- $T : P \rightarrow$ *Data-types* is a function that assigns a data type to each parameter. *Data-types* is a set of XML data types.

- *U:* $P \rightarrow$ *Units* is a function that gives the unit of measurement used for each parameter. Units is a taxonomy for measurement units.
- *R:* $P \rightarrow$ *Roles* is a function that assigns a business role to each parameter. *Roles* is a taxonomy for business roles.

Definition 5: *Qualitative composability.* We say that $OP_{ik} = (D_{ik}, M_{ik}, In_{ik}, Out_{ik}, P_{ik}, C_{ik}, Q_{ik})$ is qualitatively composable with $OP_{jl} = (D_{jl}, M_{jl}, In_{jl}, Out_{jl}, P_{jl}, C_{jl}, Q_{jl})$ if:

1. Q_{ik}.Fees $\geq Q_{jl}$.Fees; and
2. (Q_{ik}.Security = true) $\Rightarrow$ (Q_{jl}.Security = true); and
3. Q_{ik}.Privacy $\subseteq Q_{jl}$.Privacy.

Where *fees* represent the amount of money charged by the provider for using the operation. S*ecurity* represents whether the operation uses security mechanisms or not. *Privacy* is related to the privacy of the information (the operation parameters), and whether this may be divulged to a third party or not.

The rest of this chapter studies the model presented by (Medjahed et al., 2003) in "Composing Web services on the Semantic Web". The approach used in this, as shown in Figure 7.1, consists of four conceptually separate phases: *specification, matchmaking, selection, and generation.* Each of these phases is examined in some detail in the following sections and subsections.

7.2.1 The Specification Phase

This phase focuses on the specification of the required composite service. Medjahed et al. (2003) defined an XML-based language, called *Composite Service Specification Language* (CSSL) which can be used to describe a composite service. One of the essential characteristics of the CSSL, which is simple enough to provide a high-level description of composite services. Composers are required to have just a basic amount of knowledge about the services (functionalities) that they are interested in. These services can be provided by outsource agents

CSSL characteristics	Other service composition language characteristics
CSSL adopts an ontology-based model (introduced in Sect. 7.2) to cater for Semantic Web-enabled Web services.	Most of the existing languages do not consider semantic capabilities of Web services.
The CSSL specification of a composite service does not refer to any outsourced service.	In other languages composers insert references to component services in their composite service specifications.
CSSL specifications can be used as the entry point for the (semi-)automatic generation of composite services.	BPEL4WS, XLANG, and WSFL languages are used for static service composition. Other languages are partially supporting the automatic generation of composite services (semi-automatic).
CSSL defines a WSDL-like language for composite services. It extends the WSDL language to allow: (1)The description of semantic features of Web services and, (2) Specification of the control flow between composite service operations.	Some of these languages have adopted an XML-based format (IBM, 2007, Leymann, 2008, Microsoft, 2001, Florescu et al., 2003). The rest have developed their own language for service composition.

Table 7.1: Four basic differences between CSSL and others service composition languages

(service providers). In other words, they do not need to provide full technical details, such as the detailed description of the outsourced services, their interaction protocols, message format, etc. Medjahed et al. noted that there are considerable differences between CSSL and existing service composition languages (IBM, 2007, Leymann, 2008, Microsoft, 2001, Schuster et al., 2000, Casati et al., 2000, Lazcano et al., 2000, Florescu et al., 2003). Table 7.1 shows four of the basic differences.

These characteristics of CSSL make the definition process of composite services as easy as the definition of single Web services (i.e. non-composite). In addition, it supports the recursive composition of services. Since CSSL defines a WSDL-like language then a composite service can be seen as a WSDL service and therefore is used as a single basic service components for a new composition step. Figure 7.2 provides a brief overview of the main features of CSSL.

```
<service name="car broker"/>
 <category domain="brokerage">
 .....
 <binding name="SOAP"/>
 <message name="offer">
 <parameter name="price" type="float" unit="US dollar" role="extendedPrice"/>
 <parameter name="make" type="string" ...../>
 <parameter name="model" type="string" ...../>
 <pararmeter name="year" type="gYear" ...../>
 <parameter name="mileage" type="integer" ...../>
 </message>
 .....
 <operation name="receiveSpecialOffers" mode="one way"/>
 <input name="offer"/>
 <category domain="automobile dealer">
 <synonyms>
 <synonym value="car dealer"/>
 </synonyms>
 .....
 <purpose function="price-sales catalogue"/>
 .....
 <quality>
 <fees value=0/>
 .....
 </operation>
 <flow source="getPayingHistory" target="applyForFinancing">
```

Figure 7.2: CSSL specification, taken from Medjahed et al. (2003)

7.2.2 The Matching Phase

This phase is responsible for generating a composition plan that corresponds to the service specification which was developed in the previous phase by using CSSL. This phase depends on the matchmaking algorithm that was proposed by Medjahed et al. (2003). See Figure 7.3 and Figure 7.4 for more details of this. The algorithm involves searching the entire service registry which may result in generating a large number of plans. Thus, the service composers are able to control the number of generated plans by using the *nb_requested* plans parameter. To decrease the overhead of checking the compatibility of processes, the algorithm uses service interfaces (WSDL) rather than the service components. This has great effect in reducing the number of services to be accessed. In fact, the same service interface can be used by different providers, and several service implementations may have the same interface (W3C, 2001). For example, Airlines may each define service interface for booking air-flight tickets. Travel agent services would then use this interface to develop their own Web services.

The matchmaking algorithm has a number of functions (see Definition 3) that for each $op_{ik} = (D_{ik}, M_{ik}, In_{ik}, Out_{ik}, P_{ik}, C_{ik}, Q_{ik})$ of the composite services WS_i are used to find and match one or more operations $op_{jl} = (D_{jl}, M_{jl}, In_{jl}, Out_{jl}, P_{jl}, C_{jl}, Q_{jl})$ of existing service WS_j. The algorithm searches for $WS_j = (D_j, O_j, B_j, P_j, C_j)$ in order that P_{ik} and C_{ik} are compatible for at least one element of P_j and C_j, consequently (lines 8 and 9 of Fig. 7.3). Then, the algorithm checks if the interacting services are binding composable (second condition in line 8). Medjahed et al. (2003) classify Web services into communities. Communities provide a means for organising ontological classifications of the possible service space. "Each community clusters Web services based on their category. All services that have similar categories belong to the same category (purpose) community. A service may belong to different communities. The use of service communities accelerates the process of discovering relevant component services". If we presume that the category C_{ik} and WS_j's category are not compatible. Then, C_{ik} is not compatible with the services that reside in WS_j's community. Consequently the Web services from this community would not be included in the pool of discovered services.

```
(01) Input : WS_i, repository, nb_requested_plans {
(02)  nb_generated_plans = 0
(03)  matched = ∅
(04)  while nb_generated_plans ≤ nb_requested_plans do
(05)   { plan = ∅
(06)    for each operation op_ik ∈ O_i do
(07)     { found = false
(08)      for each service WS_j from repository | category_compatible(C_ik,C_j)
(09)                                  and purpose_compatible(P_ik,P_j) and (B_i ∩ B_j ≠ ∅) do
(10)       { for each operation op_jl ∈ O_j | (mode_ik and mode_jl are dual) and (op_jl ∉ matched)
(11)          if purpose_compatible(P_ik,P_jl) and category_compatible(C_ik,C_jl) and quality_composable(op_ik,op_jl)
(12)              and message_composable(in_ik,out_jl) and message_composable(in_jl,out_ik)
(13)           then { found = true
(14)                  plan = plan ∪ {(op_ik,op_jl)}
(15)                  matched = matched ∪ {op_jl}
(16)                  break }
(17)       if found then break
(18)      } /* for in line (08) */
(19)      if ¬found
(20)         then { output("no matchmaking for",op_ik)
(21)                break }
(22)     } /* for in line (06) */
(23)    if ¬found then break
(24)              else if sound(plan)
(25)                   then output(plan,ST) /* ST is a Stored Template */
(26)                   else if relevant(plan,τ_relevance) and complete(plan,τ_completness)
(27)                      then output(plan,ST,τ_relevance,τ_completness) /* Test for QoC parameters */
(28)                      else output(plan,"not sound",τ_relevance,τ_completness)
(29)   nb_generated_plans = nb_generated_plans + 1
(30)  } /* while in line (04) */ }
```

Figure 7.3: Matchmaking Algorithm, taken from Medjahed et al. (2003)

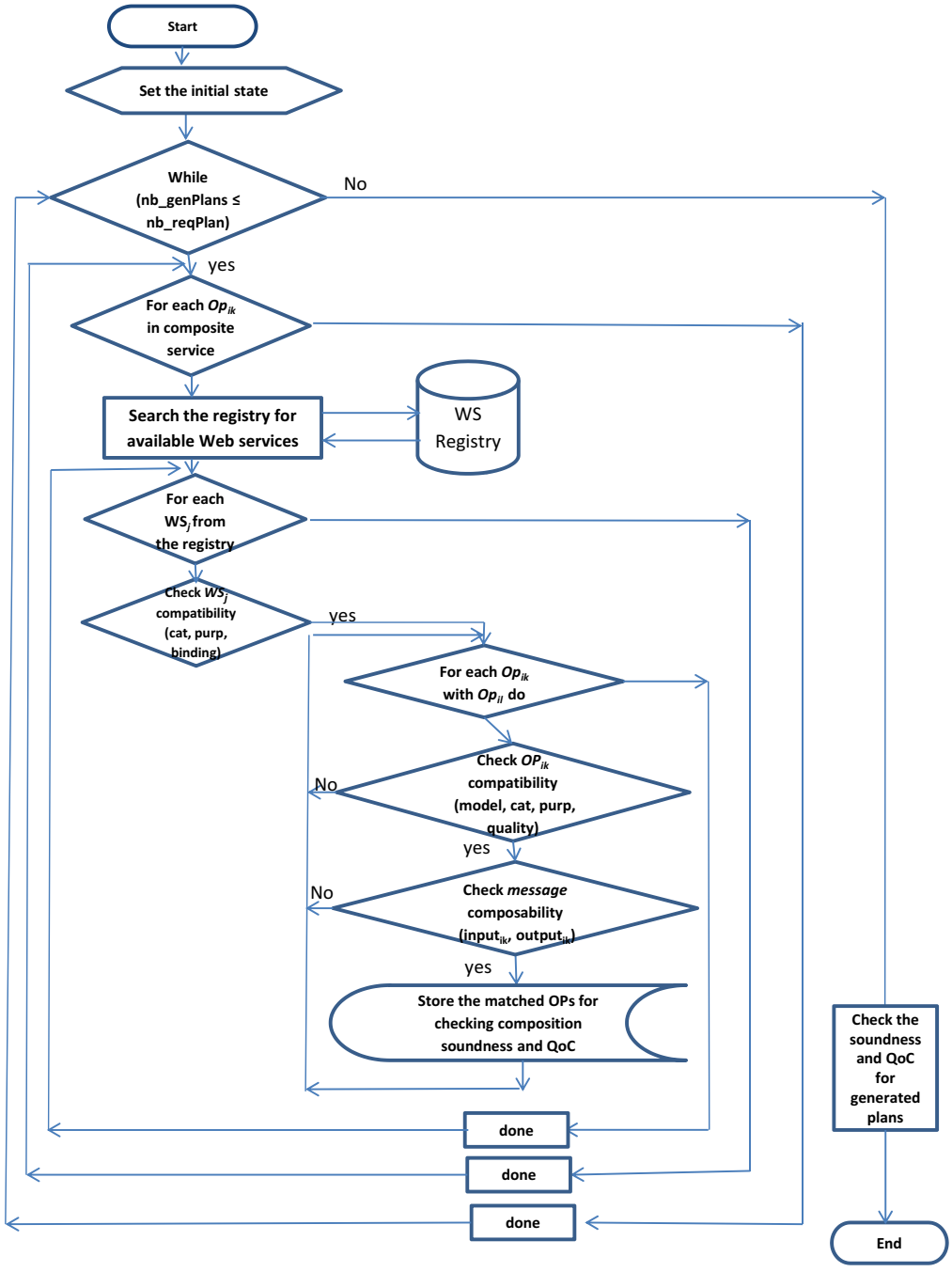


Figure 7.4: Matchmaking Algorithm flow chart diagram

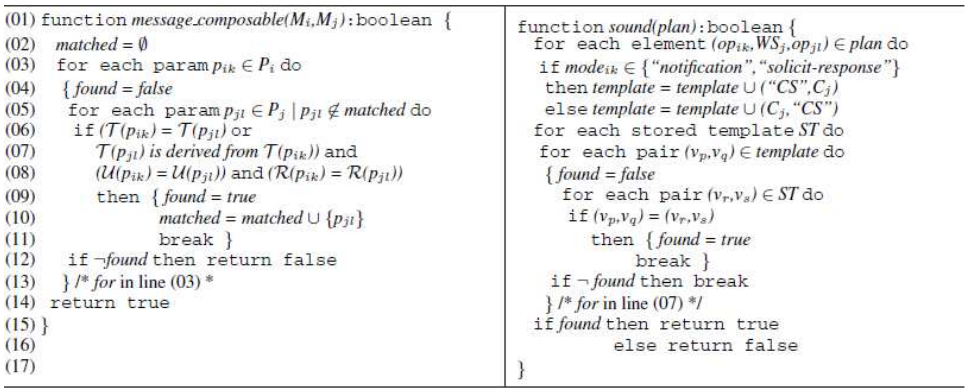


Figure 7.5: Message composability and soundness checking functions, taken from Medjahed et al. (2003)

The composition plan therefore consists of pairs of operations: the operation of the composite service and the outsources operation (op_{ik},op_{jl}). For these two operations to be coupled, the matchmaking algorithm (Fig. 7.3) checks operation mode composability (line 10), operation semantics composability (line 11), and message composability (line 12). A composition plan is generated for each iteration of the while statement (line 4). The algorithm uses a matched set to store every operation that has been "plugged into" a composite service operation (lines 10 and 15). The idea behind using this set is to avoid generating any composition plans that could be extracted from previously generated plans. Medjahed et al. (2003) give this example, "assume that the following two plans, *plan1* = {(op_{i1},op_{j1}), (op_{i2},op_{j2})} and *plan2* = {(op_{i1},op_{j3}), (op_{i2},op_{j4})} have been generated. Since *plan3* = {(op_{i1},op_{j1}), (op_{i2},op_{j4})} and *plan4* = {(op_{i1},op_{j3}), (op_{i2},op_{j2})} can be inferred from plan1 and plan2, there is no need to generate them again". The pseudo code that in lines 26–28 examines composition soundness is based on QoC parameters. There are two functions used, *complete() and relevant()*. Further details regarding the selection process and involving QoC, completeness and relevance are presented in next section.

There are a number of functions that the matchmaking algorithm uses to check the composability between two services: *purpose_ compatible(), category_ compatible(), quality_ composable(), message_ composable(), and sound()*. The first two functions *purpose_ compatible(), category_ compatible()* are used to check the purpose and category compatibility between the composite service and discovered

service, they return *true* or *false* depending on whether the purpose or category of the composite operation is compatible with the purpose or category of a discovered service operation. The *quality_ composable()* function returns *true* in the case where a composite service operation is qualitatively composable with a discovered service operation. The remaining two functions are described in Figure 7.5. For these two services, the *message_ compatible()* function returns *true* if a message M_i is message compatible with M_j, otherwise return *false* (see definition 4). The match set (line 10) has been used to allow a one-to-one mapping between M_i's and M_j's parameters. This set holds M_j's parameters that previously have been mapped to M_i's. The soundness of the generated plan is then checked by *sound()* function. When a template has been calculated for the generated plan (lines 2–5), it is compared with the stored templates (lines 6–14). The templates (composition plans) is stored and return to the composer even if they are not sound.

7.2.3 The Selection Phase

Where the matchmaking phase has finished, a number of composition plans may have been generated. The selection phase will then be used to select relevant plans. Medjahed et al. (2003) employ the following quality of composition (QoC) parameters for this: *ranking, relevance*, and *completeness*. the selection phase may also employ other QoC parameters such as *cost* and *time*. The following list presents definitions of ranking, relevance, and completeness (as defined in the original study):

- Composition ranking: this is intended to present an approximation for its "importance". For every plan, the algorithm in Table 7.5 determines its composition template CT. Presume that CT is a subgraph of a stored template ST_i. The stored template is a graph that includes all possible combinations between services that can take part in a composition and their links. The algorithm uses a function called R (R for reference) formulated on the set of stored template; $R(ST_i)$ to calculate the number of times that services with templates that are subgraphs of ST_i have been created. With respect to ST_i, the ranking of CT is the proportion of

references to ST. It is formulated as below (n is the number of stored templates):

$$\text{Ranking}(CT, STi) = \frac{R(ST_i)}{\sum_{k=1}^{n} R(ST_k)}$$

- Composition relevance: or CR for short, is intended to present an approximation of the 'soundness' of a composition. The algorithm uses this parameter to make comparison between edges of a composition template, CT, with the edges of a stored template ST_i. $CR(CT, ST_i)$ is the ratio of CT's edges that occur in ST_i. It is formulated as below (E and E_i are the edges of CT and ST_i , respectively) which gives the percentage of the number of edges that in the CT are also in the ST_i:

$$CR(CT, ST_i) = \frac{|E \bigcap E_i|}{|E|}$$

- Composition completeness: or CC for short, represents the percentage of composite service operations that are composable with component service operations. Use of the CC parameter allows generation of plans regardless to how "fully" the composite service is composable with available outsourced services. Depending on the experience of service composers the value of CC is estimated. Medjahed et al. (2003) said that "Indeed, if the value CC is relatively low (e.g., 25%), the algorithm might return plans in which 75% of the composite service operations are not composable with component service operations. In this case, composers may need to change their specification (e.g., data types) so that the desired service can deal with other services' features". Below is the formula that describes the CC parameter for a composite service WS_i :

$$CC(WS_i) = \frac{|Composable(O_i)|}{|O_{i|}}$$

where $Composable(O_i) = \{op_{ik} \in O_i \mid \exists\, WS_j\ \exists\, op_{jl} \in Oj$ so that op_{ik} is syntactically and semantically composable with $op_{jl}\}$.

Composition plans are arranged and returned based on their ranking. Plans that have highest ranking are returned first. This requires that a ranking coefficient is provided for each stored template. Composers identify two thresholds $\tau_{relevance}$ and $\tau_{completness}$ which are related to the relevance and completeness parameters, respectively. Plans are returned to composers if their relevance and completeness are greater than their respective thresholds. The parameters for QoC may be identified inside a CSSL specification, so that the "best" plans are automatically selected and returned to users.

7.2.4 The Generation Phase

Generation is the last phase in the approach. It starts when the matchmaking and selection processes have been accomplished. Its role is to generate a detailed description of a composite service. This description is composed of the list of "outsourced services, the mappings between the composite service and the component service operations, the mappings between messages and parameters, and the flow of control and data between component services" Medjahed et al. (2003). This phase has two important elements, which are customisation and extensibility. Customisation indicates the ability to produce composite service descriptions in different languages, such as BPEL4WS (Business Process Execution Language for Web Services) IBM (2007), WSFL (Web Services Flow Language) Leymann (2008), and XLANG Microsoft (2001). A service composer chooses the "target" composition language in their CSSL specification. Extensibility means that the approach can be extended to include additional composition languages. In fact the structure of composition plans is in an abstract form which can be adopted at an intermediate level between CSSL specifications and most existing Web service composition languages. This phase was not a part of the empirical implementation of the original study, therefore, it's not considered to be a part of our replication study.

7.3 Performance Evaluation

The purpose of our replication study has two aims. The first aim, is to demonstrate that we can use simulation to replicate the empirical work (experiment) that is presented in (Medjahed et al., 2003), which mainly focused on the matchmaking phase. The replication therefore consists of building a simulation model that applies the logic of the original study. Our hypothesis is that the simulation model produces similar results to the results of the original study. We therefore set the same goal as used the original experiment to our experiment, which is to assess the scalability of matchmaking algorithm, this is, the capability for generating plans for a large number of service interfaces. We focus on the matchmaking phase because it is the empirical element of the Medjahed et al. study. Because it is the phase that needs to deal with a large number of Web services to examine composability rules. In other word, the goal of their experiment was to evaluate the effectiveness and speed of the matchmaking algorithm. Another goal is to assess the role of the selection phase (QoC parameters) in decreasing the number of generated plans. The second aim of our study, is based on the success of the first aim which is used a proof of validation of our simulation model. So this can then be used for further investigation into the service composition process. Figure 7.6 describes the relationship between the original study and our replication.

7.3.1 The Experimental Set-up

The experimental set-up consists of two phases: a *developing a simulation program* phase and a *generating Simulated data_ sets* phase. Two software packages have been used for performing the activities of these phases: MySQL Workbench 5.2 CE and Java NetBeans IDE 7.1.1, both of them running under the Windows 7 operating system. MySQL is used as the UDDI registry to store information about composite service specifications as presented in the WSDL service interfaces. Java is used to develop a program for generating data-sets and to build the simulator program for the matchmaking process. We ran our simulation program on a Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz, 2M cache (2 processors) RAM 256 GB, and under the 64-bit Windows 7 Oper-

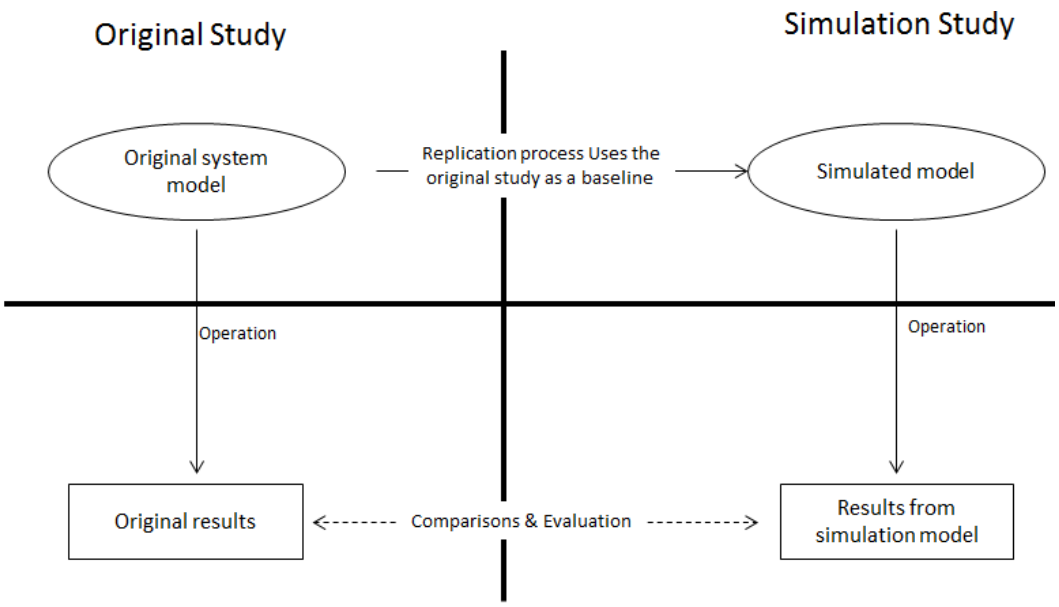


Figure 7.6: Close replication description

ating System. Our simulator program is able to generate a large number of data-sets. Furthermore, the generated data-sets have Web semantic description characteristics that are not available in existing Web service description data.

7.3.1.1 The Use of Simulated Data-sets

Pickard et al. (2001) used simulated data-sets to compare data analysis techniques used for software cost modelling. They observed that "Simulation was used to create data-sets with a known underlying model and with non-Normal characteristics that are frequently found in software data-sets: skewness, unstable variance, and outliers and combinations of these characteristics" (Pickard et al., 2001). The most significant result of their study was to demonstrate the value of using simulation as a technique for evaluating different data analysis techniques under controlled conditions.

The literature review showed a lack of usage of any real data-sets with the characteristics that we needed for our replication study and that could be used as a 'baseline'. In addition, the original study mentioned that the current Web service technologies still do not enable the semantics-driven data access and

processing (Medjahed et al., 2003). Therefore, they generated their own data-sets that allowed the use of semantic-driven data to examine their approach. Similarly, we needed to generate the necessary simulated data-sets in order to replicate and study the use of their Composing Web Services model on the Semantic Web. The experiment focused on the matchmaking process because it is the empirical part of the original study.

The generation of the simulated data-sets followed the same steps that were described by (Medjahed et al., 2003). We used a relational database (MySQL see Fig. 7.7) to store Web service information represented as a WSDL document (service interface). Five tables were used to store this information, where these were: service, binding protocol, operation, message and part tables. These five tables represent the essential information that extends WSDL.

1. The Service table is used to store the basic information about a Web service such as name, category, purpose, etc.

2. The Binding table was used to store information about the protocols of each Web service.

3. For each Web Service in the service table, we used an Operation table to store information about its operations. Two type of information were stored, functional and nun-functional properties, which were very important for the matchmaking and selection processes. Each operation had one or two messages depending on its mode. If the operational modewas “one-way” or “notification” that means it had one message either “input” or “output”, respectively. If the operation mode is "request-response" or "solicit-response" that means it had two messages “input” and “output”.

4. The Message table was used to store information about these “input” and “output” messages.

5. Each message consists of several parts. These parts represent the input/output parameters that pass between the services. Each part was described by number of characteristics such name, data-type, unit and business role. The Parts table was used to store this information.

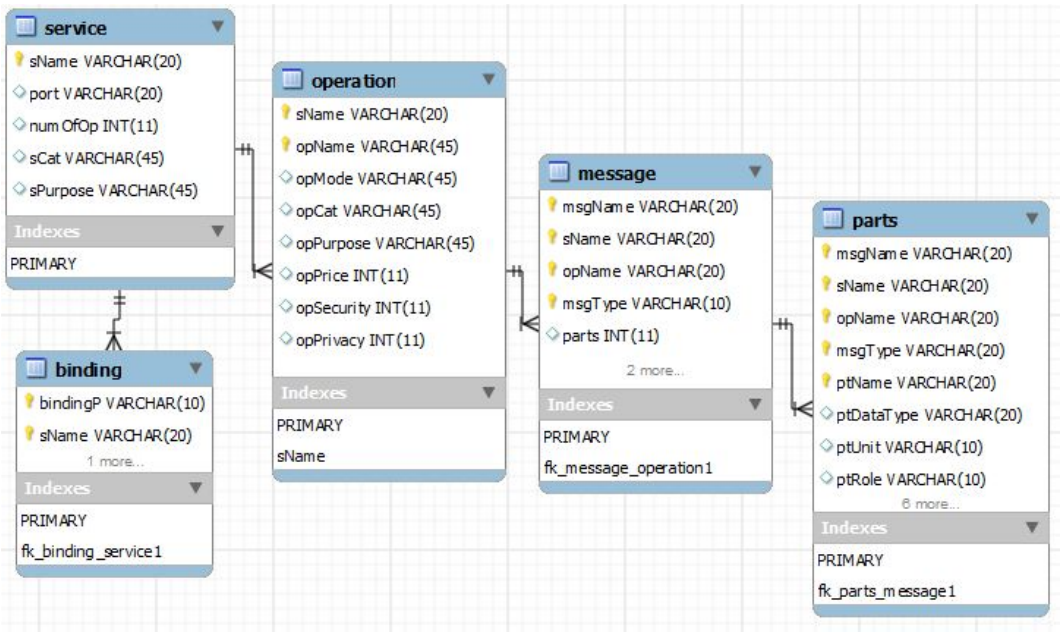


Figure 7.7: Data-Sets Tables using MySQL Database EER Model

7.3.1.2 The Simulation Program

There are number of frameworks that approach the Web service composition dynamically (Shuiguang et al., 2004, Patil et al., 2004, Liang et al., 2007, Liu et al., 2010). The matchmaking and selection processes are the essential part of their success. To implement these two processes in our simulation framework, we have used semantic Web and Quality of Service strategies, respectively. According to the outcomes from the SLR, we found that the semantic Web is the most major technique used for the matchmaking process (Medjahed et al., 2003, Cheung et al., 2004, Mokhtar et al., 2007, Segev and Toch, 2009). The Quality of the Service (QoS) has been widely adopted in numbers of frameworks for achieving a business process plan dynamically with preferable qualities that requested by the WS consumers (Gu et al., 2004, Chen et al., 2006, Jun et al., 2007). We have built a simulation framework by applying the ideas from these strategies with compliance of the Service Oriented Architecture (SOA). This chapter we mainly focuses on the use of semantic Web for matchmaking process. For the use of QoS model for selection process, we will provide more details in the next chapter.

The aim of the Simulation Framework is to assist the WS consumers to formulate their Web service composition. The framework enables the WS consumers to generate various composition plans and choose the one that meets its quality preferences. The implementation substantially depends on java NetBeans and OO techniques. The OO techniques are useful to apply some promising ideas, such as using objects to hold the WS information. This information is used to hold the matched Web service properties such as service name, service description, purposes, categories and binding protocols. Information about service's operation such as operation name, operation semantic and mode. Furthermore, we used objects to hold information about messages and message's parameters (See Figure 7.8 for more details). The simulation framework implements the matchmaking algorithms (7.3 & 7.5), as we mentioned earlier. The following are the essential processes in the simulation framework:

- The simulation starts with random select of a composite service, forms a pool of composite service (10 composite services). Then, based on the description of required services, the simulation searches the database to find service candidates.

- Checking services composability: The purpose of this process is to check if the founded service interface is composable with the composite service. This means that the service interface and the composite service have the same category and purpose.

- Checking operation composability: If the service interface is composable, then, each operation in the composite service is checked with the service interface operations. This process involves checking operation's mode and semantic.

- Checking message composability: If the two operations of the composite service and the service interface are composable, then, this process checks the composability of the messages that is passed between these operations. The process involves checking message's parameters data-type, unit and business role. The accomplishment of these checking composability processes results on generating a composable plan.

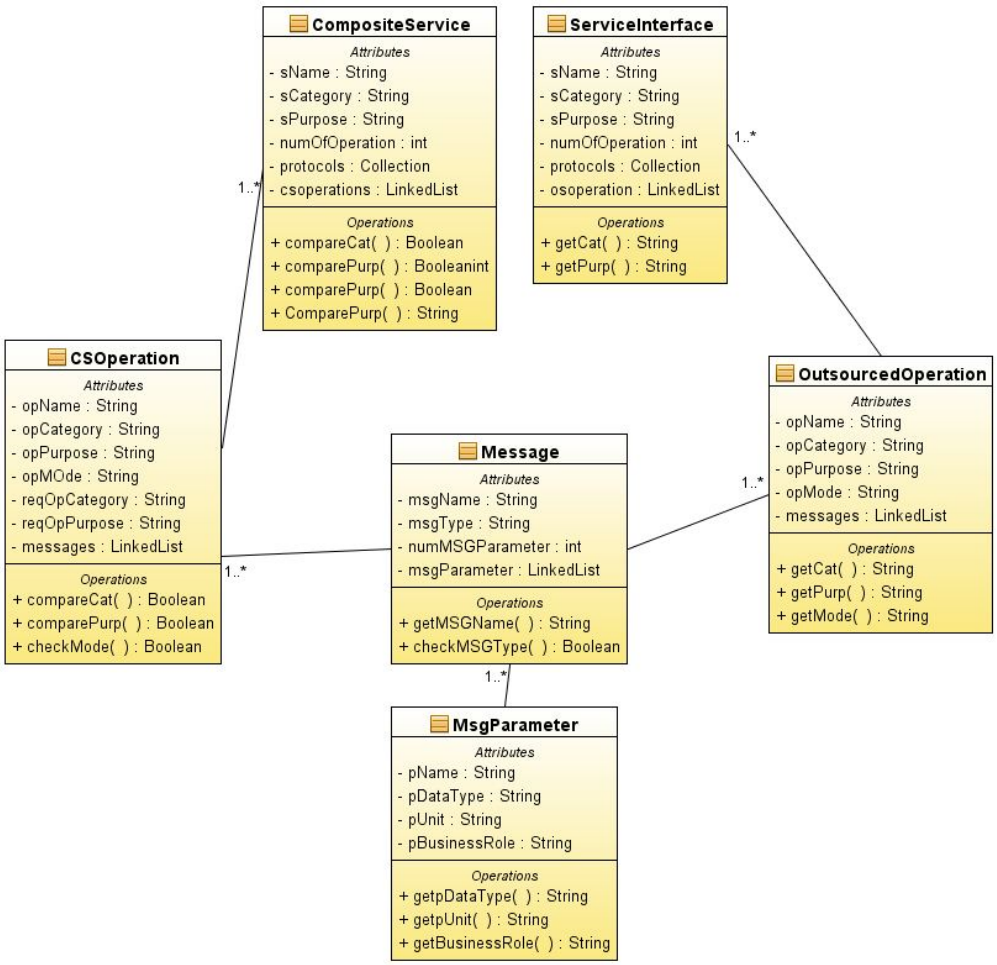


Figure 7.8: Simulation Framework: Class diagram

- Checking composition soundness: The purpose of this process is to check each pair of operations in the generated plan with the stored templates. Stored templates are defined as sets of operations which can achieve a specific business goal. These templates either defined by experts in the field or have been learned by the system from previous composition tasks.

7.3.2 The Experimental Parameters (Independent and Dependent Variables)

In a similar manner to the original study, our simulator programme allows users to control the choices of range and statistical models for several of the quant-

Variable	Range
Service interfaces	3000–30000
Composite services	100–1000
Operations per service	10–50
Parameters per message	50–100
Requested plans	50–100
Stored templates	100–500
Vertices per stored template	10–20

Table 7.2: Independent variables

itative attributes in the experiment. These attributes include operations per service, messages and parts per message. In our replication experiment, we followed the same method of randomly generating values for the independent variables as was used by Medjahed et al. (2003). We used a uniform distribution to randomly generate service interfaces, composite services and stored templates. For our experiment, we altered different parameters including the number of service interfaces, composite services, operation per services, parameters per message, requested plans, stored templates and nodes per templates (see Table 7.2). These parameters represent the independent variables for this experiment. The following list describes these steps:

- First set the number of services (from 3,000 to 30,000 with an iteration range of 3,000).
- For each service, then generate a category (1 out of 50), its binding protocols and the number of operations.
- For each operation, generate the mode, purpose (1 out of 100), category and quality.
- Finally, for each input and output message, we randomly generated the number of parameters, data type, unit and business role of each parameter (1 out of 37 built-in data types).

The experiment examines how these parameters affect the number of generated plans and plan generation time (dependent variables). Through the experiment, for every alteration of the independent variables we measured the number of

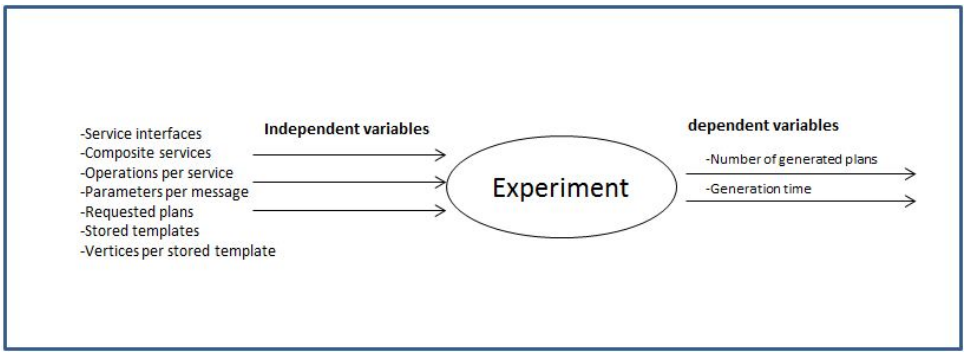


Figure 7.9: Independent variable vs. Dependent variables

generated plans and the time. In addition, we also interested to study the time needed for checking compatibility of Web services, operations, messages and soundness. We conducted two experiments to study the relation between the independent variables and dependent variable. The following subsections give more details about these. Figure 7.9 gives an overview of how the experiment processes the inputs (independent variables) to generate the outputs (dependent variables).

7.3.3 Experimental Procedure and Results

In this replication study, we followed the same experimental procedure as the original study (Medjahed et al., 2003). Similarly, our experiment has examined two aspects. For the first aspect, assessing the generation time for composition plans, three execution times are examined. The first execution time is computed as the time that is spent to check mode, binding and operation semantics composability. The second execution time is computed as the time that is spent to check message composability. The third execution time is computed as the time that is spent to check composition soundness. The results of the replication study are similar to the results of original study in terms of having a same pattern/profile. Figure 7.10 shows both results from simulation in the left and the original study in the right. The following points give interpretations for the results of the replication study and make comparisons with the results of the original study.

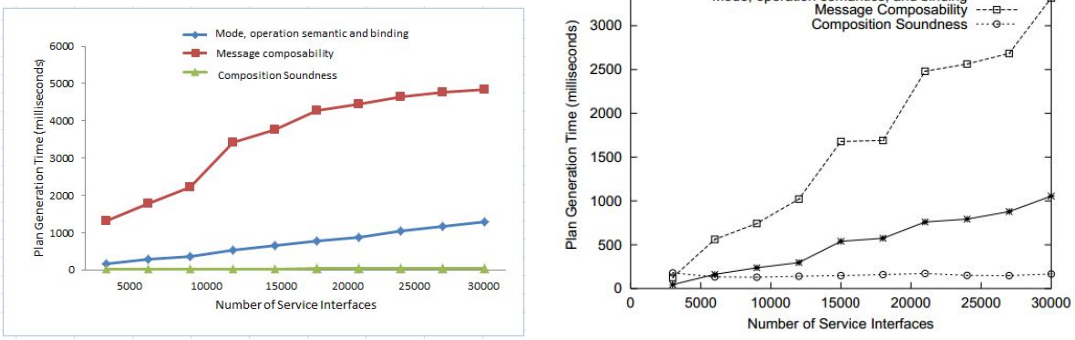
7.3.3.1 Replication Interpretation: First Experiment Results

1. Computing for the first time involves checking operation category, purpose, mode and binding protocols processes which are less CPU-intensive. It increases linearly when the number of interfaces is increased. From the comparison of the results, we found that the first time has similar pattern for both the original experiment and the replication. The first time is represented by the mode, operation semantic and bind composability line in Figure 7.10.

2. We found that the process of checking message composability is the most time consuming through overall matchmaking/generation plan processes (Fig. 7.10). Indeed, the second time demands a high computation time for each matched pair of operations of composite service and Web service respectively, in order to compare each parameters of their input and output messages. From the comparison of the results, again we found that the second time show a similar pattern in both the original experiment and the replication. The second time is represented by the message composability line in Figure 7.10.

3. The third time, which checks the soundness of generated plans, has the lowest time of the generation processes. Indeed, the process of checking the composability of syntactic and operation semantic of composite services for all service interfaces in the services registry (3000 to 30000 service interfaces) consumes more time compared to the process of checking the soundness of generated composition plans (100 to 500 stored template). Therefore, this explains relatively a stable line which represents the time for checking composability soundness process. Additionally, the third time has similar pattern in both the original experiment and the replication. The third time is represented by the composition soundness line in Figure 7.10.

However, the plan generation time does not take into account the inquiry time for the service registry as well as the stored template repository. Our simulation results are not exactly identical values as the results of the original study, but



a) Replication study results b) Original study results

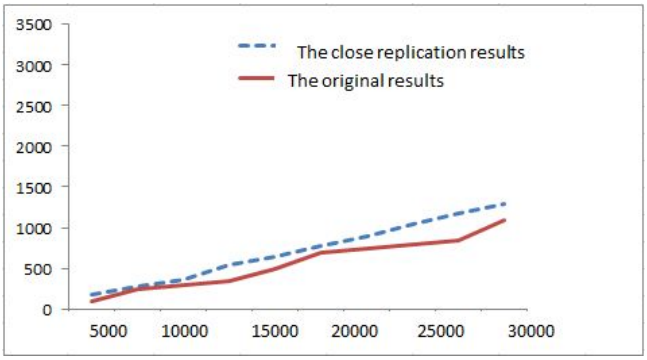
Figure 7.10: Plan generation time: Replication study vs. Original study

both results have similar pattern. We attribute this to the use of different datasets and different programming structures in the two studies. For instance, we have used a MySQL database to store the service interfaces information while the original study used a text file to store the service interfaces information. Both studies used the Java as programming language and its OO programming paradigm, but it may be different objects have been used in these studies.

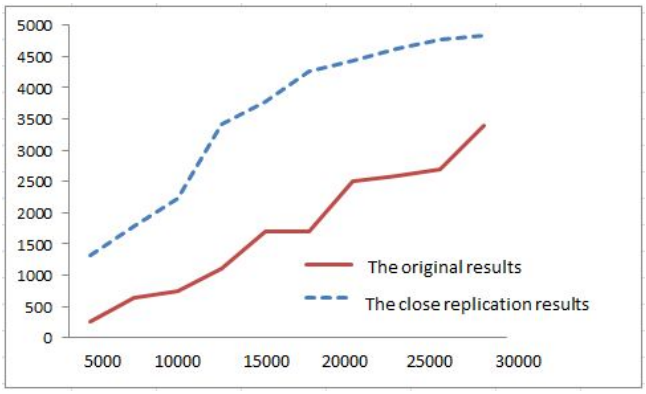
Furthermore, we separated the generation time of composition plans, as presented in Figure 7.10, into three figures to make the comparison more clear. Although we did not have the real values of the results from the original study, but we were able to extract these values from their result diagram. Figure 7.11.a represents the time for checking message compasability. Figure 7.11.b represents the time for checking mode, operation semantic, and binding compasability. And Figure 7.11.c represents the time for checking composition soundness.

7.3.3.2 Replication Interpretation: Second Experiment Results

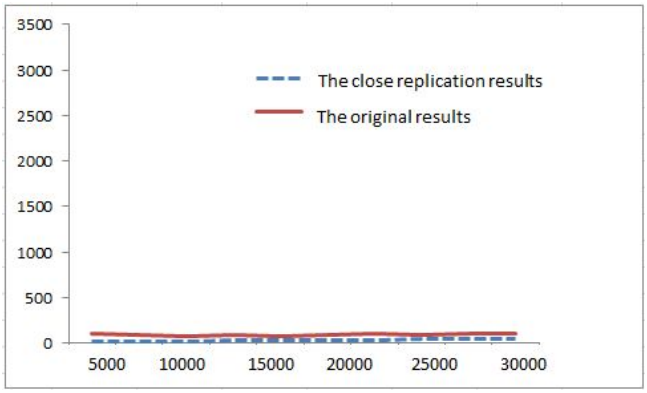
For the second aspect, we have tested the Composition Completeness (CC) ratio for its effect on the number of generated plans. Similar to the baseline study, we preformed experiments for CC = 33% and CC = 66%. Figure 7.12 shows that our experiments and the baseline results have similar profiles, with results from our simulation in the left and the original study in the right. For CC = 33%, the number of generated plans is higher than CC = 66%. Indeed, when the CC



a) Checking message compasability

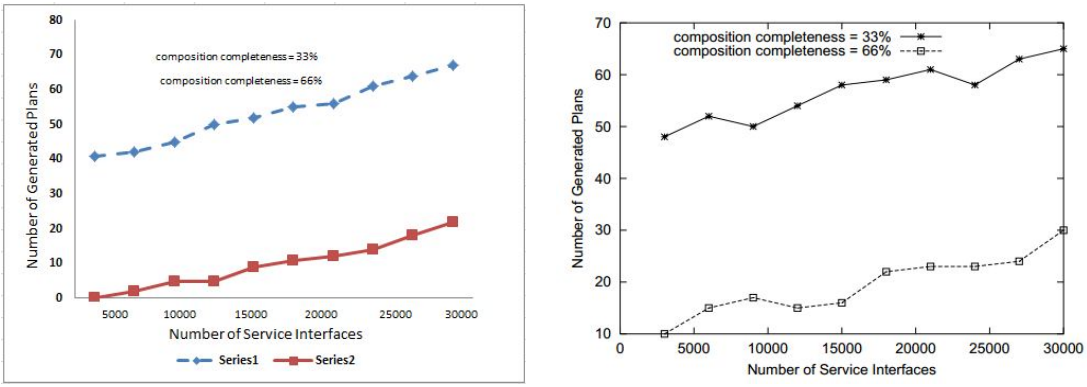


b) Checking Mode, Operation semantic, and Binding compasability



c) Checking composition soundness

Figure 7.11: Comparison between the three execution times of generation time



a) Replication study results b) Original study results

Figure 7.12: Number of generated plans

ratio is 33%, plans are generated if at least 33% of composite service operations are composable with outsourced service operations. On the other hand, when the CC ratio is 66%, plans are generated if at least 66% of composite service operations are composable with outsourced service operations. In addition, our results show that the number of generated plans for CC = 33% is, on average, greater than 50 which is equal to the minimum number of requested plans. This is interpreted as that plans have been generated for almost every particular composite service. In contrast, for CC = 66%, the number of generated plans is at most equal to 30 which is less than the minimum number of requested plans. This is interpreted as for some composite services, no plan has been generated. This supports the findings of the baseline study about the impact of CC ratio on the number of generated plans. Both studies indicate that a comparatively low value of CC ratio generates more plans, but then each plan contains a small number of composable operations. On the other hand, although a high value of CC ratio generates a smaller number of plans, each plan have more composable operations.

So far the outcome of our replication study, we found that the results of the replication and the original experiment show similar patterns. The similarity between both studies is due to our close replication study that has followed the same experimental procedure of the original study, and our simulation model applied the same logic as the original study. In other words, the replication was a close form, and we have been able to increase confidence in the original

results. In the light of the results, we can claim that our simulation model show suitably similar behaviour of the original study.

7.4 Summary

This chapter has described an external replication of a study that was presented by (Medjahed et al., 2003) in "Composing Web services on the Semantic Web". The replication study empirically evaluated the result of the original study and made comparisons between the results of the replication and original study, in order to investigate whether the outcomes of the original study can be repeated or not. The results of the replication study have similar profiles to those of the original study, which confirms the validity of the original study results, and at the same time can used as a proof of validity of our simulation model.

Chapter 8

Web Service Selection

8.1 Introduction

This chapter introduces the use of semantic Web and QoS models for the selection process. The model used extends the description of Web services by adding the non-functional properties of the Web services. This model is then integrated with the simulation framework that we have presented in the previous chapter to investigate the effect of extending the selection process in this way. The QoS model is used to define sets of quality criteria that can be used to rank similar Web services in order to select one of them according to service consumer preferences. We perform a number of experiments to compare between the performance of the QoS model and a ranking mechanism which are presented by Medjahed et al. (2003).

8.2 Quality attributes: Definition and Measurement

Fenton and Pfleeger (1998) state that "from a measurement perspective, we must be able to defined quality in terms of specific software attributes of interest to the user. That is, we want to know to measure the extent to which these attributes are present in our software product". The knowledge about

the software attributes of interest to the user allows us to determine quality attributes in a measurable form. The definition and measurement of an external quality attribute also have to be determined with consideration as to how the product relates to its environment. For example, in the case that the product is software code, its reliability (defined in terms of the probability of failure-free operation) is an external attribute. This attribute relies on both the user and the machine environment. When we examine a software code as our product and we look at an external attribute that depends on the user's preferences, we surely deal with an attribute that is related to a particular view of quality (i.e., a quality attribute). Accordingly, the attributes that we consider in this chapter are related to some ideas about software quality and particularly the Web service quality. The aim of this chapter is to identify small number of relevant external attributes and determine their measurement.

8.3 Background: What is the Selection Process?

The analysis of the outcomes of the literature review in Chapter 4 shows that many proposed approaches offer promising solutions for Web service composition. In particular, the Semantic Web and QoS strategies are the major approaches in terms of the number of studies published, and how extensively they have been studied in the context of service composition (see the analysis of the literature review for more details, Chapter 4). Currently, the number of Web services that can provide the same functionality has been increasing (Yu et al., 2007, Badr et al., 2008). For example, for such purpose as booking flight Web services, weather forecast Web services, etc. Therefore, service consumers need somehow to be able to differentiate between these services. The generic composition framework discusses in Chapter 5 shows that service composition has many essential processes (steps). The *selection* process is one of these processes. This step is mainly concerned with using the non-functional properties of the Web service to select an appropriate WS. Generally, the outcome of the discovery (matchmaking) step results in a pool of candidate WSs that have a high degree of functional similarity for performing a specific task. By using same measure for the minimal similarity degree that is acceptable to the service

consumer, the selection step then excludes all those WSs that fail to meet this condition. The service composer performs this step if there is more than one WS that can achieve a specific task, but it offers different services, at different costs and quality levels. The selection process then involves performing a comparison between the remaining WS candidates, and re-ranking them, based upon the user's preferences regarding the non-functional properties of the WS (such as response time, cost, availability, etc.). Examples are given by Zeng et al. (2003) and Liu et al. (2004b).

The literature review shows that all QoS strategy papers have addressed the selection process. In fact, the QoS strategy approach is mainly focused on the selection process. Their solutions are based on using the non-functional properties in order to generate an appropriate service composition plan that meets the service consumer's requirement. These solutions employ similar techniques that are based on a QoS model and a computation mechanism for service composition. Four studies categorised under the semantic Web and ontology strategy proposed specific mechanisms for service selection (Medjahed et al., 2003, Cheung et al., 2004, Younas et al., 2006, Mokhtar et al., 2007). Their mechanisms are based on performing a semantic match of the service capabilities with the needs of the user task, then using a Web service selection function based on the QoS specification to select the most appropriate WS among those services that matched the user request. Other studies used different strategies. One study used a Knowledge-based strategy to guide the process of WS selection by giving advice about suitable WS for a specific task depending on the user's preference data (Xiaogao and Xiaopeng, 2006). Another study used a software agent (user agent) to interact with service providers (provider agents) in selecting the most appropriate service based on a single quality attribute (cost) (Zakaria et al., 2004).

8.4 Semantic Web and Non-functional Properties

This section introduces the ideas of adopting a semantic Web model, as described in (Medjahed et al., 2003) in the previous chapter. Medjahed et al.

identified three qualitative properties for operation: *fees, security and privacy.* In addition, other qualities may also be included such as time, availability and latency. Medjahed et al. set fixed criteria for these properties. The fees of operation are given in dollars, which represents the amount of money paid to use the service. Security and privacy are considered as important features of a Web service (Rezgui et al., 2002). Businesses usually store, collect, process and share information from interested users who may have different preferences regarding the security and privacy of their information. Security and privacy are very important in many areas, for example, e-banking systems, NHS systems, etc., where people are sensitive about access to their information. Here, the security property is represented by a Boolean value used to indicate whether the operation's messages are securely exchanged between clients and service providers. The privacy property concerns any information that a service's consumer is not willing to have disclosed to other parties rather than the service provider. Medjahed et al. define the notion of operation quality as follows:

Definition 1 - Quality. The quality of an operation op_{ik} is defined by a tuple ($Fees_{ik}$, $Security_{ik}$, $Privacy_{ik}$). $Fees_{ik}$ is the amount in dollar needed to execute op_{ik}. $Security_{ik}$ is a Boolean that specifies whether op_{ik}'s messages are securely exchanged. $Privacy_{ik}$ is the set of input and output parameters that are not disclosed to external entities. Here, i represents the composite service number, and k represents the Web service number that can achieve a specific task.

Due to the dynamic and heterogeneous characteristics of the Web service environment, and in order to improve this definition to accommodate other qualities, we can use semantic Web to describe each quality of a Web service. Furthermore, this also helps in automating the selection process for Web services. The description of qualities (properties) is similar to the description of input and output parameters used in (Medjahed et al., 2003). Each quality is described by *name*, *data type*, *unit* and *synonyms*. *Name* and *synonyms* are used to identify the quality. *Data-type* is used to define the data type of the quality. Compatible data-types belong to the same category, e.g., numerical data-types,such as integer, float, double, etc., fall under one category. When comparing between the qualities of two or more WSs, that is, their data types in all one category

and do not have the same detailed form, then all data types are converted to the data type that has the biggest memory space. For example numerical category data types are transferred into *double* data-type. *Unit* is used to indicate the measurement unit for the quality criterion. Here, we use the same standard measurement units (length, area, weight, currency, etc.) as used by Medjahed et al. (2003) to assign values to quality units. Each quality of an operation is described by a tuple (name, data-type, unit and synonym). To make comparisons between two or more operations we use the following definition:

Definition 2 - For each quality of an operation op_{ik}, its quality criterion q_{ik} is described by the tuple ($name_{ik}$, $datatype_{ik}$, $unit_{ik}$ and $synonym_{ik}$). In order to make comparison between two or more operations, op_{jl}'s qualities q_{jl}s, the comparison mechanism matches between these qualities and makes the required conversion.

1. $q_{ik}.name = q_{jl}.name$ or $q_{ik}.name \in q_{jl}.synonym$ or vice versa; and
2. $q_{ik}.datatype = q_{jl}.datatype$ or $(q_{ik}.datatype \ \& \ q_{jl}.datatype) \in Datatype - Category$,
3. $q_{ik}.unit = q_{jl}.unit$ or $f(q_{ik}.unit) = g(q_{jl}.unit) = U$, where $U \equiv$ a measurement unit that is specified by the service composer.

For example, to compare the amount of money that a service consumer should pay to use a service, one WS might name it as fees and other name it as price. These two properties are equivalent since they are synonymous. If the $q_{ik}.datatype$ is an *integer* and $q_{jl}.datatype$ is a *float* then these two values are converted into the *double* data-type. Finally, if $q_{ik}.unit$ is in pounds and $q_{jl}.unit$ is in Euros, then these two units should be converted into same currency unit that is specified by the service consumer. Therefore, this definition can be used to automate the selection process and to include other qualities.

8.5 The Quality of Service (QoS) Model

As previously noted, the number of Web services that can provide the same functionalities is increasing (Yu et al., 2007, Badr et al., 2008). Therefore, ser-

vice consumers need some mechanism that can be used to differentiate between these services. The non-functional properties (QoS criteria) can have a significant impact on the expectations of service consumers, that can be used to make a choice between similar services. Consequently, Web service descriptions can be strengthened by adding the QoS criteria (Chaari et al., 2008). Many studies proposed approaches for QoS of Web service that address generic quality criteria such as price, execution time, reliability and availability (Gu et al., 2003, Nie et al., 2006). These generic qualities may not be adequate for some domains. Other studies have used a QoS model that contains a set of quality criteria for Web Services (Zeng et al., 2003, Liu et al., 2004b). These QoS models are more comprehensive in terms of number of qualities that can be used for most of the service domains. Furthermore, in the majority of current approaches, advertising the value for a QoS is the responsibility of service providers. Also, the model offers an interface for service providers to access these values, which are subject to manipulation by them. In addition to this, the service providers may not always advertise their QoS information in a "fair" way, for example, when quoting values for execution duration and reliability. Some approaches mainly use active monitoring mechanisms to collect QoS information, which generates an overhead due to QoS values that are checked for a large number of Web services. On the other hand, other approaches depend on third parties to collect QoS information (rate or accredit) for a particular service provider, which is an expensive and static mechanism. Our model integrates the semantic definition from the previous section which extends the study of Medjahed et al. (2003) (see the replication study chapter), and the QoS models proposed in (Zeng et al., 2003, Liu et al., 2004b). The aim of this integration is to build a comprehensive and extensible QoS model in order to enhance the selection process. Furthermore, we can compare our approach with the ranking mechanism proposed by Medjahed et al. for the selection process to determine its effectiveness.

The model has three aspects, which are discussed as follows:

- **Extensible and Flexible QoS Model**: Having one general standard QoS model that can be applied to all Web services is not a practical idea (Liu et al., 2004b). Web services are registered under different domains.

Each domain has set of different QoS criteria that are of concern to their consumers. For example, the QoS that are of concern to an e-payment Web services, such as security and privacy, are different from the QoS that are of concern to e-auction Web services, such as execution time and availability. Therefore, we define an extensible and flexible QoS model that contains domain specific criteria as well as common (generic) criteria. This approach has the ability to allow new domain specific criteria to be included and can be used to evaluate the QoS of Web services with no need to alter the fundamental computation model.

- **Preference-oriented Service Ranking:** The selection process for Web services can be driven by the service consumers' preference (Liu et al., 2004b). Different consumers may use different QoS preferences and priorities to select a Web service. It is necessary for the QoS model to have a mechanism to represent the QoS criteria from the perspective of the service consumers' preference. For instance, one service consumer may be concerned about obtaining low service prices, regardless of the time it takes to invoke the service, while another consumer may be more concerned about service reliability. This means that the criterion such as cancellation or compensation fees may be considered more important than the price and time. Another service consumer may be concerned about the response time, as time is considered to be critical for his application. Consequently, a QoS model should provide a mechanism for service consumers to state their QoS criteria preferences before the selection process takes place.

- **Fair and open QoS Model:** The purpose of having a fair and open model is to build a reliable QoS model that is useful to both service provider and consumer. A reliable model can also ensure that the set of QoS criteria for each domain is collected in fair manner (Liu et al., 2004b). The collection of QoS criteria can be through various means such as properties that are advertised by service providers, service execution monitoring mechanisms and service consumer feedback, depending on the quality criterion. For example, the amount of money for using the service is set by the service provider (cost, fees, price, etc.). While the time that

elapses from invoking a service to get a response back is calculated by the service monitoring mechanism (execution duration, response time, etc.), the service reputation is based on feedback from service consumers based upon their experiences of using a service. To avoid the manipulation of the QoS criteria, Liu et al. (2004b) built a policing mechanism that requires each service consumer to have an authentic identity and password to fill in the values for QoS criteria of the service that has been consumed. Moreover, this identity and password have to be approved by the service providers at the invocation of the service, which guarantees that only the actual consumer is allowed to give feedback. Besides this, to be absolutely fair, service providers can review those criteria and can enhance the QoS of their Web service if they wish to do so. Furthermore, service providers can update certain QoS information such as execution price and compensation rate at any time. Providers can then access the service registry to review how their QoS is ranked when comparing with other service providers.

These three aspects of a QoS model are essential for the service selection process. They make a QoS model suitable and benefitial for all service consumers. This model also has the particular property that requires service consumers to send their feedback to a registry each time when they consume a service. This extra effort of sending a feedback by the consumer is not an overhead task. Consequently, the QoS for a particular service is being regulated by all service consumers. All service consumers have access to the registry to look up for most-updated QoS of listed providers. Also, service providers are able to access the registry to check out and modify their services at any time. "With such an open and dynamic definition of QoS, a provider can operate its Web services to give its end-users the best user experience" Liu et al. (2004b).

8.6 Computation of the Web Service Qualities (QoS) Model

The experimental results from the previous chapter show a range of composition plans generated for a composite service. These plans are able to achieve the

composite service objective/purpose with different QoS values. Only one plan is then selected according to the three QoS criteria, which are *ranking*, *relevance*, and *completeness*. The selection process checks the generated plans for their relevance and completeness and then returns them to the composer in rank order. The plan with highest rank is returned first. In this chapter we extend the selection process by an extensible QoS model which contains a number of other QoS criteria. The extensible QoS model includes five generic quality criteria (Zeng et al., 2003). These generic quality criteria are usually found in all Web services. Moreover, these quality criteria include qualities that are advertised by service provider. Those qualities are collected through execution monitoring mechanism, and result from service consumers' experiences by using a specific Web service. The following list defines these quality criteria as:

- **Execution price (service fees)**. This represents the amount of money that a service consumer needs to pay the service provider in order to invoke/use a service operation. For a given operation op of a service s, this quality criterion is denoted as $q_{price}(s; op)$. The price for using a service operation is set by the service providers when advertising their services, or they offer means to inquire about it. The service consumer determines the currency unit to which all service fees units have to be converted.

- **Execution duration (response time)**. The execution duration time is the elapsed time when a request to a service has been made and its corresponding response has been received. For a given operation op of a service s, this quality criterion is denoted as $q_{du}(s; op)$. The execution duration is computed by using the expression: $q_{du}(s; op) = T_{process}(s; op) + T_{trans}(s; op)$, which indicate that the execution duration is the sum of the operation execution time (processing) $T_{process}(s; op)$ and the transmission time $T_{trans}(s; op)$. The provider advertises the processing time of his services in the registry or provide methods to inquire about it. The value of the transmission time is calculated from the previous execution history of the service operation, that is, $T_{trans}(s; op) = \frac{\sum_{i=1}^{n} T_i(s; op)}{n}$, where $T_i(s; op)$ is the previous i's transmission time, and n is the number of execution times recorded in the past. Execution duration is measured

in milliseconds.

- **Reputation.** The reputation of service s is denoted as $q_{rep}(s)$ which is used as a measure of a service trustworthiness. This quality depends on service consumers' opinions and the satisfaction of their experience of using a service. Different consumers may have different opinions of their experience with same service. Therefore, the representative value of the reputation is described as the average ranking of overall values that are supplied by the end users. The average ranking is computed as $q_{rep}(s) = \frac{\sum_{i=1}^{n} R_i}{n}$, where R_i is the reputation ranking value that is given by the end user on a service. And$n$ is the number of times the service has been rated/ranked. Generally, service applications provide the end users with a range of values to rank the service reputation. For example, in Amazon.com, the range is [0, 5].

- **Reliability**. Reliability of service s is denoted as $q_{rel}(s)$ which represents the capacity for maintaining the service and service quality. The number of failures per month or year represents a measure of reliability of a Web service. In other words, reliability refers to the assured and ordered delivery for messages being sent and received by service consumers and service providers. The value of the reliability is calculated from the previous execution history of the service operation. That is, $q_{rel}(S) = N_c(S)/K$, where $N_c(S)$ represents the number of times that the service s has been successfully delivered within a specified period of time, and K represents the total number of invocations that have been made of the service s.

- **Availability:** Availability of service s is denoted as $q_{av}(s)$ which represents whether the Web service is present or ready for immediate use. Availability can be measured as the probability that a service is available. Larger values mean the service is always ready to be used while smaller values indicate uncertainty about whether or not the service will be available at a particular time. The service availability value is calculated using the following formula: $q_{av}(s) = \frac{T_a(s)}{\theta}$, where T_a is the interval of time (in seconds) that a service s has been available over the last θ seconds (the service community is responsible for setting the constant value of θ). Different applications may have different values of θ. For example, for

an application domain, such as a stock exchange, where services are frequently accessed, a small value of θ gives an appropriate approximation for the availability of services. On the other hand, for an application, such as an online shopping store, where services are not accessed frequently, a larger value is more appropriate. This model requires the Web services to provide notification to the system relating to their current states, i.e., available, unavailable, (Zeng et al., 2003).

The following vector describes the five quality criteria of a Web service as:

$$q(s) = (q_{price}(s),\ q_{du}(s),\ q_{rep}(s),\ q_{av}(s),\ q_{rel}(s)) \qquad (1)$$

8.7 Quality Criteria for Composite Services

In a similar way to know how they are used for a single service (service component), the aforementioned quality criteria can also be applied to assess the QoS of a composite service. Zeng et al. (2003) describe in Table 8.1 a computation formula for the QoS of a composite service execution plan. The plan is described as p = $\{< t_1;\ s_{i1} >, < t_2;\ s_{i2} >, \ ...\ , < t_N\ ;\ s_{iN} >\}$. A short explanation of each criterion's contribution to the computation formula is given as follows:

- *Execution price*: The execution price $Q_{price}(p)$ of an execution plan p is the sum of each service s_i's execution price $q_{price}(s_i; op_i)$ for those services s_i taking part in the plan p.
- *Execution duration*: Zeng et al. adopted a Critical Path Algorithm (CPA) to compute the execution duration $Q_{du}(p)$ of an execution plan p. Particularly, the CPA is used to compute the critical execution path of execution plan P, described as a project digraph. The *critical path* of a plan is the longest path, starting from the initial state, and going to the final state of plan p, and is the count of the number of service components. The aggregation of the duration of each service component in the critical execution path of plan p then represents the execution duration for plan p.

Criteria	Aggregation formulas
Price	$Q_{price}(p) = \sum_{i=1}^{N} q_{price}(s_i, op_i)$
Duration	$Q_{du}(p) = CPA(q_{du}(s_1, op_1), \cdots, q_{du}(s_N, op_N)$
Reputation	$Q_{rep}(p) = \frac{1}{N} \sum_{i=1}^{N} q_{rep}(s_i)$
Reliability	$Q_{rel}(p) = \prod_{i=1}^{N} (e^{q_{rel}(s_i) * z_i})$
Availability	$Q_{av}(p) = \prod_{i=1}^{N} (e^{q_{av}(s_i) * z_i})$

Table 8.1: Aggregation formulas for computation the QoS of composition plan (Zeng et al., 2003)

- *Reputation*: The reputation $Q_{rep}(p)$ of an execution plan p is the average of each service s_i's reputation $q_{rep}(s_i)$ in the execution plan p.
- *Reliability*: Reliability $Q_{rel}(p)$ of an execution plan p is the product of $e^{q_{rel}(s_i)*z_i}$. In the aggregation formula, z_i has two values; 1 or 0. z_i equals 1 if s_i is a critical service in the execution plan p, otherwise z_i equals 0. For example, if s_i is not a critical service, then $z_i = 0$, and $e^{q_{rel}(s_i)*z_i}$=1 which means the reliability of service s_i does not affect the value of the execution plan's reliability.
- *Availability*: The availability $Q_{av}(p)$ of an execution plan p is a product of $e^{q_{av}(s_i)*z_i}$, where $e^{q_{rel}(s_i)}$ is service s_i's availability. Factor z_i is treated in the same way as in the previous definition.

These definitions of aggregation formulas are used to produce the quality vector of a composite service's execution plan. The vector consists of five quality criteria, which is described as follows:

$$Q(p) = (Q_{price}(p),\ Q_{du}(p),\ Q_{rep}(p),\ Q_{rel}(p),\ Q_{av}(p)) \qquad (2)$$

8.8 Service Selection Process

As mentioned in the generic life-cycle model for Web service composition, service selection is an essential and independent process which comes after the matchmaking process. The matchmaking process results in a number of plans

that can achieve a composite service goal. The role of the selection process is to select the optimal plan according to the chosen criteria. We have used a selection mechanism that is based on the QoS model that is described in the previous sections. The QoS model involves five quality criteria. Therefore, the decision involved in selecting an optimal plan needs to be based on some combination of these quality criteria. Zeng et al. (2003) have stated that "the selection of a service is based on a selection policy involving parameters of the request, the characteristics of the members, the history of past executions, and the status of ongoing executions". In the following subsections, we will present the computational approach used here for computing the QoS of a composite service.

8.8.1 Combination of Multiple Quality Criteria

The aim of our QoS model is to represent the QoS criteria of a Web service from the perspective of the service consumers' preference. The representation involves a combination of multiple quality criteria that our model uses to describe the qualities of a Web service. In order to combine the quality criteria in our QoS model, we have adopted the *Multi-Attribute Utility Theory* (MAUT) as described by Bolinger et al. (1978). Many studies in software engineering have used the MAUT as an evaluation mechanism when there are multiple attributes being used to describe a product (Chang and Ho, 2010, Chang, 2008, Ross et al., 2009, Hong-yu and Ying, 2010). MAUT is an evaluation scheme which is also used by consumer communities for evaluation of products. As the basis for MAUT, Schäfer (2001) defined the total evaluation $v(x)$ of an product x as a weighted addition of its evaluation with respect to its relevant value attributes. For example, a flight journey can be evaluated on the basis of price, duration, luggage allowance, cabin type, the number of stop-over, etc. The total evaluation is defined by the following overall value function:

$$v(x) = \sum_{i=1}^{n} w_i v_i(x) \qquad (3)$$

Here, $v_i(x)$ represents the evaluation of the attribute on the i-th value dimension d_i and w_i represent the weight that is used for resolving the effect of the

i-th value dimension on the total/overall evaluation (also called the relative importance of a dimension[1]), n represents the number of criteria/dimensions, and we use the constructive that $\sum_{i=1}^{n} w_i = 1$.

8.8.2 Selecting an Optimal Execution Plan

In general, the matchmaking process identifies several execution plans. Each plan consists of one or more execution paths. These execution paths are either simple paths, which means they are executed in sequential order, or complex paths, which means their execution depends on conditional operations that can drive the execution through different branches (Zeng et al., 2003). In this study, we focus on the execution plans that have only one simple execution path. The execution plan consists of many tasks ,t_j, and each task, t_j that can be achieved by a set of matched services, S_j. The quality criteria for each service, s_{ij}, in this set is presented as a quality vector (see equation 1). Depending on the available Web services, and as a result of selecting a Web service for each task in an execution path, a set of plans, P , is be generated:

$$P = \{p_1,\ p_2,\ \ldots\ ,\ p_n\} \qquad (4)$$

Where n represents the number of execution plans. When a set of execution plans has been generated, the next step is to use a mechanism to select an optimal plan from them. For the selection of execution plan, rather than calculating the quality vector of an individual Web service, each execution plan's overall service quality vector needs to be calculated.

Once the quality vector for each plan has been collected, then all the execution plans' quality vectors are aggregated together which gives a matrix $\mathbb{Q}$, where each row represents an execution plan's quality vector.

[1]The relative importance of a dimension also expresses the relevance of a dimension for the overall evaluation (Schäfer, 2001)

$$\mathbb{Q} = \begin{pmatrix} Q_{1,1} & Q_{1,2} & ... & Q_{1,5} \\ Q_{2,1} & Q_{2,2} & ... & Q_{2,5} \\ \vdots & \vdots & \vdots & \vdots \\ Q_{n,1} & Q_{n,2} & ... & Q_{n,5} \end{pmatrix} \qquad (5)$$

To select the optimal plan, a Simple Additive Weighting (SAW) method is applied to matrix $\mathbb{Q}$ (Zeng et al., 2003). SAW is one of the implementations of MAUT method. Basically, there are two computation phases in using SAW:

8.8.2.1 Scaling Phase Using Min-Max Normalisation

The Web service quality criteria have different values for the service consumers. For some of the quality criteria (called negative criteria), the lower value they hold, is better for the service consumers, such price and execution duration. For other quality criteria (called positive criteria), the higher value they hold is better for the service consumers, such as availability and reliability. Zeng et al., therefore, defined two formula/equations to scale matrix $\mathbb{Q}$ by using min-max normalisation. Min-max normalisation is a process of a linear transformation of an original data that is measured in measurement units (for example, package per second, or centigrade degree) (Han, 2005). The data is transferred into a value between 0.0 and 1.0. For the negative criteria, the minimal value is set to 1.0 and the maximal value is set to 0.0. And for positive criteria, the minimal value is set to 0.0 and the maximal value is set to 1.0. This offers a simple way to make comparison between values that are measured by different units/scale (for example, price and duration). Formula 5 is used to scale the negative criteria, while formula 6 is used to scale the positive criteria.

$$V_{i,j} = \begin{cases} \frac{Q_j^{max} - Q_{i,j}}{Q_j^{max} - Q_j^{min}}, & if \;\; Q_j^{max} - Q_j^{min} \neq 0 \\ 1 & if \;\; Q_j^{max} - Q_j^{min} = 0 \end{cases} \qquad j = 1, 2 \qquad (6)$$

$$V_{i,j} = \begin{cases} \frac{Q_{i,j} - Q_j^{min}}{Q_j^{max} - Q_j^{min}}, & if \;\; Q_j^{max} - Q_j^{min} \neq 0 \\ 1 & if \;\; Q_j^{max} - Q_j^{min} = 0 \end{cases} \qquad j = 3, 4, 5 \qquad (7)$$

In these two formula (6,7), Q_j^{max} represents the maximal value of a quality criterion j in matrix $\mathbb{Q}$, i.e. $Q_j^{max} = Max(Q_{i,j}), 1 \leq i \leq n$, and Q_j^{min} rep-

resents the minimal value of a quality criterion j in matrix $\mathbb{Q}$, i.e. $Q_j^{min} = Min(Q_{i,j}),1 \leq i \leq n$.

However, these two values $Q_j^{max}$and Q_j^{min} can be computed without generating all possible execution plans. For example, the maximum execution price (i.e. Q_{price}^{max}) of all execution plans is computed by selecting the most expensive Web service of each task and summing them together gives the maximum execution price Q_{price}^{max}. Similarly, the minimum execution duration (i.e. Q_{du}^{min}) is computed by selecting the Web service that has the shortest execution duration time of each task to use a critical path algorithm to compute the Q_{du}^{min}. The computation cost for the $Q_j^{max}$and Q_j^{min} values is polynomial.

By applying the scale phase on matrix $\mathbb{Q}$ we, therefore, obtain the following matrix:

$$\mathbb{Q}' = \begin{pmatrix} V_{1,1} & V_{1,2} & ... & V_{1,5} \\ V_{2,1} & V_{2,2} & ... & V_{2,5} \\ \vdots & \vdots & \vdots & \vdots \\ V_{n,1} & V_{n,2} & ... & V_{n,5} \end{pmatrix} \qquad (8)$$

8.8.2.2 Weighting Phase

For each execution plan, its overall quality score is computed by the following formula:

$Score(p_i) = \sum_{i=1}^{5}(V_{i,j} * W_j)$ (9)

where $W_j \in [0,1]$ and $\sum_{j=1}^{5} W_j = 1$. W_j represents the weight of each criterion. In formula (9), the value of W_j is adjusted by the end users according to their preferences of QoS (i.e. balance the impact of the different criteria) to be able to select the desired execution plan. Based on the value of $score(p_i)$, the system will choose the execution plan that has the highest value (i.e. $max(score(p_i))$). In case there are more than one execution plan with the same value of $score(p_i)$ then the system randomly will select one of them.

8.9 Validation

In this section, we present the implementation of the QoS driven selection of service as described in Zeng et al.. The purpose of the implementation is to replace the selection mechanism described in Medjahed et al. (2003). Then, we conduct some experiments in order to compare these two selection mechanisms in terms of overall QoS of the composite service. The experimental results are used to evaluate the proposed approach.

8.9.1 Implementation

In this chapter, we adopt a QoS model related to the QoS of execution plans. This is used to facilitate the selection process to choose the most optimal execution plan that meets the service consumer preference. The proposed model consists of five qualities criteria and their computation technique is implemented and integrated in the simulator programme that we have already developed for close replication study. The aim of the integration is to replace the selection mechanism which is based on ranking criterion (Medjahed et al., 2003). For further details of the replaced mechanism are discussed in close replication chapter.

This is a brief reminder of what we have discussed about our simulator programme in the previous chapter. Our simulator applies some of basic operations of the Service Oriented Architecture (SOA). Mainly, we focus on matchmaking and selection processes. The implementation is based upon two software packages, Java NetBeans and MySQL database. We extended our simulator program to include a selection mechanism based on the QoS model that we described earlier. The selection mechanism applies the computation formulas to calculate the final QoS of composite service by applying the scale and weighing phases. The programme uses a MySQL database as the Service Registry. Services could be registered under many categories. To enhance the registration process—the registry uses taxonomy for naming and organising Web services into groups that share similar functionalities. These groups create classifications which assist the service providers to register their services under the suitable categories. Each category, besides its name and description has many

keywords which offer more options which can be used to inquire the registry for the appropriate taxonomy that may contain the intended Web Service. As mentioned in the description of the close replication, the database contains other tables used to store the description's information of the WS that have been advertised in the WSDL file. In order to conduct our experiment on service selection process, we adopted a QoS model to present the non-functional properties of the WS. The model applies execution quality criteria, such as execution price, execution duration, reliability, reputation and availability, in scale or percentage, therefore, a table was added to store these items of information.

8.9.2 Experimentation

In this section, we present two experimental results. The first experiment compares the computation method of QoS model with the ranking mechanism that is used in the close replication study (see the previous chapter). The comparison is based on using the QoS metrics of the selected plans for both mechanisms. The second experiment compares the computation time of these two mechanisms. Each time, we randomly generate several composite services with different numbers of tasks. For each task, the number of selected services are varied according to the number of services (from 3,000 to 30,000 with iteration range of 3,000). Therefore, when the range get bigger, then the number of generated plans gets bigger with every iteration.

8.9.2.1 QoS Metrics of the Selected Plans

The purpose of this experiment is to compare the QoS values from using the QoS model approach with those from the ranking approach. The comparison is done in two ways. First, we measure the QoS of the selected plan as a single value ($Score(p_i)$) for both mechanisms. In addition, we compare the selected plans by the five individual qualities. In this experiment, we add the five qualities of the QoS model into the description of each Web service operation. The values of these five qualities are randomly generated. We use our simulator programme to generate ten different outcomes of composite service plans through the matchmaking phase. For each outcomes, the number of

generated plans ranges over 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 plans. We repeat this for ten different composite services. Each time, we recod the QoS values as a single score value and also as individual quality values. Because we obtain very similar experimental results for all generated composite services, we only present an one set of results from these composite services (see Table 8.2). We put other four results diagrams in appendix (C).

We can conclude from the table that when using QoS model approach to compute the overall QoS of a composite service, this will have a higher score than when using a ranking approach. The last two columns show the score values for the ranking and QoS models respectively. Similarly, the last row represents the average of each quality criterion of the composite service plan, showing that the average values from using the QoS model is better than from the ranking approach. Although there are some instances where quality criteria using ranking mechanism are better than the QoS model quality criteria, this is always less than half of the quality criteria (no more than two out of five). For example, in instance 1, the execution price for the ranking mechanism is better than for the QoS model, in instance 2 and the reliability value for the ranking mechanism is better than for the QoS model, in instance 3, Both the execution price and reliability values for the ranking mechanism are better than for the QoS model., However, the overall quality score is better for the QoS model than for the ranking mechanism. Therefore, we can conclude that the use of the QoS model mechanism gives better results than the use of the ranking mechanism in terms of the overall quality criteria of the resulting composite service.

All simulations show that the QoS approach produced better plans that have higher quality score than those produced by the Ranking method. This can be seen in the last 2 columns (QoS, Ranking) in Table 8.2.

8.9.2.2 Computation Time

The other purpose of this experiment is to investigate the computation time that the simulator needs in order to select an optimal plan. The investigation again involves both the QoS model and the ranking mechanisms. We record the computation time for selecting the optimal plan using the generated composite plans which range over 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100., and repeat

	QoS model					Overall Score	
Instance No	Execution Price	Execution duration	Reputation	Reliability	Availability	QoS	Ranking
1	8.07	13.6108	5	0.6904	0.9084	0.671	0.5745
2	5.6064	13.5622	5	0.5941	0.8291	0.6978	0.332
3	6.5211	20.5639	5	0.747	0.9201	0.6772	0.5604
4	5.9883	12.2555	5	0.9844	0.9004	0.8365	0.4293
5	1.0906	26.6638	5	0.7766	0.9998	0.8257	0.6572
6	1.1747	11.6709	3	0.967	0.6497	0.7437	0.2583
7	9.3326	14.7487	4	0.9985	0.8582	0.7167	0.4397
8	1.8679	37.4421	5	0.9989	0.9132	0.8124	0.4425
9	1.0832	16.6388	4	0.979	0.9815	0.905	0.6624
10	3.4099	20.1144	4	0.8104	0.9611	0.7687	0.658
Average	**4.41447**	**18.72711**	**4.5**	**0.85463**	**0.89215**	**0.76547**	**0.50143**

Table 8.2: Comparison between Ranking and QoS model techniques

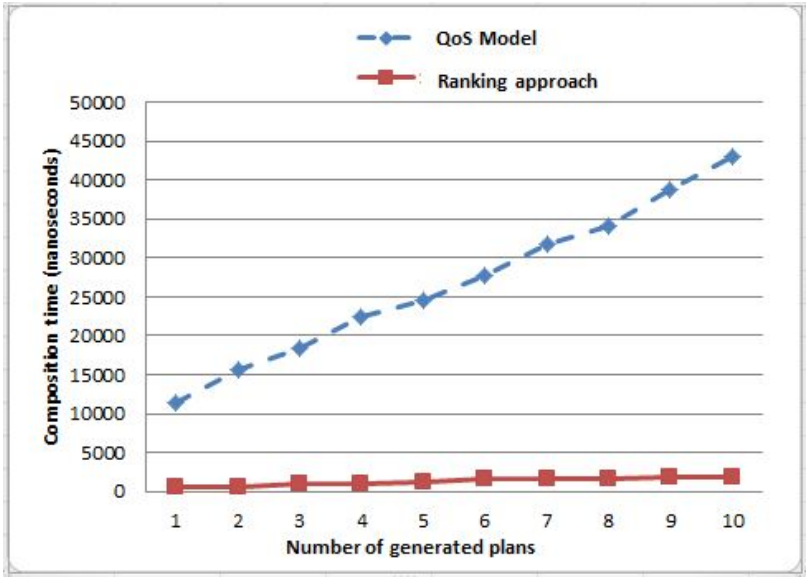


Figure 8.1: The computation time for the QoS model vs. the Ranking approach

this ten times over these plans. Again, because we obtain much the same experimental results for all generated composite services, we only present a one experiment results from these composite services (see Fig 8.1). We put other four results diagrams in appendix (C).

Figure 8.1 shows the computation time for both the QoS model and the ranking approaches. We notice that the computation time is consistently higher when using the QoS model for selecting an optimal plan. For QoS model, the computation time is linearly increasing when the number of generated plans is increased. The computation time when using the ranking only increases very slightly when the number of generated plans is increased.

In close replication chapter, we test the Composite Completeness (CC) ratio for its effect on the number of generated plans. We use two CC ratio values, 33% and 66% respectively. For 33%, the number of generated plans range between 40 and 70 , and for 66%, the number of generated plans range between 0 and 30 generated plans. In order to evaluate the performance of the QoS model and the ranking approaches, we compare the generation times for 30, 50 and the 70 plans. Table 8.3 shows this comparison.

However, as been shown in Figure 8.1 and Table 8.3, the computation time for both approaches is not very expensive as all values are less than one millisecond.

Number of Generated plans	Generation time in nanoseconds		Difference
	QoS model	Ranking	
30	17416	933	**16483**
50	23325	1244	**22081**
70	29545	1555	**26435**

Table 8.3: Computation time difference between QoS model and Ranking approach

Therefore, the significance of the plan generation time depends on how this time is critical for a particular system.

8.10 Summary

In this chapter we introduced two mechanisms to enhance the selection process for Web service composition. We extended the description of Web service by using semantic Web to add the description of non-functional properties to the description of a Web service. Then, we introduced a QoS model that has a set of five quality criteria as non-functional properties. This QoS model can be extended by the use of a semantic Web to include other quality criteria. Based on these quality criteria, we presented a number of computation formulas to compute the QoS of composite service. The QoS model and its computation are implemented and integrated into the matchmaking programme that we implemented in the previous chapter. The purpose of the integration is to replace the ranking mechanism that is used in the original study. We conducted a number of experiments to compare the performance of the QoS model with the ranking mechanism. The experiments showed that the QoS model has better results than the ranking mechanism in terms of overall QoS for a composite service. The drawback of the QoS approach is that it can hide major defects in a plan—for example a plan may contain attributes that are the highest ranking in 4 attributes and the lowest in one and still be chosen. If the lowest is the Reliability attribute then an unreliable plan will be chosen. This may be of course due to the weight all being set to 1. In exchange, the QoS model takes a longer time to select a composite service plan, although this time is still very small (less than one millisecond). Therefore, we conclude that the QoS model generally has a better performance than the ranking mechanism.

Chapter 9

Discussion

9.1 Introduction

In this chapter we examine the likely threats to the validity for both the SLR and the experiment in turn. Then discuss the issues presented by our categorisation of the available experiences of service composition. This chapter also discusses the experiences from other studies that have used simulation to study DWSC approaches.

9.2 Threats to Validity: The SLR

Given that the practices of systematic reviews are generally well established, we would argue that *construct* validity is not an issue here. Additionally, since the aim is not to generalise our results beyond a specific topic of service composition, we do not need to consider *external* validity.

For *internal* validity, there are two issues related to the processes that we follow need to be discussed:

- Did we find all of the relevant papers?
- Did we correctly classify those we found?

Previous experience from conducting mapping studies and systematic reviews suggests that the use of four electronic databases is usually sufficient to ensure that our search is likely to find the majority of papers (anecdotal experience suggests that up to 10% more can be found by snow-balling from the references of those found, although this is usually the most effective when the search terms are less well established than is the case here). In addition, to reflect the emergent nature of the topic, we consider e including databases that would access major conferences. While our prototyping of possible search strings successfully identify the set of 'known' papers we use as a check, there remains the possibility that some papers have been missed in our search because they used terms such as 'orchestration' rather than 'composition'. Thus, while our prototyping suggests that there are relatively few of these, however, there remains the possibility that some relevant papers may have been missed. While not crucial for a mapping study, this still needs to be noted.

The second question is rather harder to address, since it implicitly includes both the inclusion / exclusion phase, and also the categorisation. Both of these tasks are performed mainly by one person (the author) with reference to the supervisor when appropriate. Where the quality of the abstract for a paper is made it becomes difficult to decide about inclusion / exclusion, we consult the main paper. The subsequent task of categorisation involves reading the complete paper, and we are reasonably confident that after some discussion, we would categorise these correctly/ , if not always exactly as the original authors might have done.

9.3 Threats to Validity: The Experiments

Through the Mapping study we found a number of studies that employed simulation to study the dynamic Web service composition (Shepperd and Kadoda, 2001, Pfahl, 2007, Kalasapur et al., 2006). Moreover, in Chapter 6, we gave some background about simulation, followed by a brief survey of papers that employed simulation for software engineering. We analyse these based on the guidelines that have been proposed by Barton (2010). The outcomes of the analysis shows that simulation has been effectively used in several studies in

software engineering (see section 6.2.2). Therefore, we are confident that making use of simulation is an appropriate form to use for our experiment. We follow both the guidelines and the outcomes for developing our simulation framework. Zeng et al. (2003) presented a similar study to our work in terms of using same QoS model and MAUT approach. Their study proposed a global planning approach for optimally selecting WS at run time (using linear programming techniques). They conducted experiments in order to compare their approach with another technique (local selection approach) using the implemented prototype system. Their results showed that the global approach effectively selected high quality execution plan.

In our experiment we first use the simulation framework to perform a close replication for the study that is presented by Medjahed et al. (2003) "Composing Web services on the Semantic Web". Although the implementation of the simulation is completely new, the results are very similar, giving confidence in the simulation itself. We then perform another experiment based on using a QoS model for Web service selection process. This experiment is an extension of the close replication study. As external validity, we do not aim to generalise our results beyond using simulation to study the effect of using semantic Web and QoS models on the process of Web service composition. However, some issues still remain to be addressed with regard to our simulation framework in the following subsections:

9.3.1 The Construction of the Simulation

There are some question related to construction of our simulation.

- Does the simulation deliver what we want in terms of experiment design?
- Using it, are we able to answer the RQ?
- What conclusions can be drawn from the results?

Simulation offers some scope for making comparisons between different techniques. Therefore, it provides the exploratory framework needed to give a focus for more systematic knowledge gathering. Chapters 7 and 8, described the steps

that we carried out to build our simulation framework, such as an investigation of the previous literature that studied service composition. The results show that this approach can deliver what we want in terms of experiment design. In performing the close replication, we find that the results from both the close replication and original studies have similar patterns which could be considered as a proof of validity of our simulation framework. Using this, we perform another experiment for our extension study. The results show that using the QoS model for selecting an optimal plan is better than using the ranking mechanism of the original study in terms of selecting a composite plan that has highest quality score.

9.3.2 Internal Threats to Validity

We reviewed two papers that employed simulated data-sets for their studies. Pickard et al. (2001) identified some analysis techniques and used simulated data-sets to investigate which techniques could be trusted to analyse software datasets reliably, regardless of certain problems. However, importantly, the result demonstrate the value of simulation as a method for evaluating different data analysis techniques under controlled conditions. Rothermel et al. (1999) used data-sets to investigate test case prioritisation techniques. Their data-sets and analysis have been used to provide insight into the effectiveness of test case prioritisation. Both studies demonstrated the viability of using simulated datasets for investigation software techniques.

In Chapters 7 and 8 we mentioned that for our study we need to use data-sets with certain properties. In addition, the literature review showed a lack of real data-sets that can be used to study service composition. Therefore, we also use simulation to generate data-sets with the properties that we need for our experiment. We use the normal distribution to generate the values of the data-sets. Using this, we are able to draw some conclusions from the results which answer our research questions But we still cannot generalise this conclusion. That is because we may need to use additional data-sets that have various kind of characteristics found in software data-sets (such as skewness, unstable variance and outliers) (Pickard et al., 2001).

9.4 Classification Issues for the SLR

We originally begin the mapping study with the expectation that we would find a corpus of empirical work in the form of experiments, case studies and surveys that can form the basis for a fuller review, as occurs with some other topics in software engineering. However, we realise that the relative immaturity (and incompleteness) of the service paradigm, the variety of possible approaches to service composition, and the difficulty of performing comparative studies across these, constrain the scope to perform systematic evaluations of the form we originally expected to find. We therefore broadened our searching to include all forms of 'experience' related to service composition.

A consequence of the very limited scale of systematic study through empirical forms is that the papers found offer little in the way of 'comparative' evaluation of the ideas being investigated in part, because of the lack of any real baseline for comparison. (More generally, the services community also seems to lack any good and widely accepted 'service scenarios' that can be used as benchmarks). Indeed, if we return to the framework proposed by Wieringa and Heerkens (2006) that was discussed earlier, then we can certainly confirm that the studies that are found are very largely *world* solutions rather than *knowledge* ones.

9.5 Interpreting the Outcomes: The SLR

Any mapping study addresses the papers that are published within a 'window' in terms of time and so inevitably, any conclusions need to be interpreted as applying to that period. For a topic such as this, where the relevant technology is evolving relatively rapidly, the combined lag in reporting primary studies and the secondary study may mean that more recent form (such as RESTful) may not be represented. While this does not invalidate our conclusions, it does place a constraint upon them. However, since REST is largely concerned with issues related to data transfer, we suggest that it is unlikely to significantly influence composition processes or any comparisons between them.

Regardless of the underlying mechanisms, the results from our study suggest that dynamic composition is still largely a 'laboratory' issue that is the primary

of interest researchers. Certainly, we find no indication that any of these forms are suitable for commercial adoption, with the systems being evaluated almost entirely being prototypes and with evaluation often being at the *concept implementation* level of "we built it and it worked".

One obvious question raises is whether dynamic composition of services is seen as relevant by industry and commerce, at least, in the forms that it presently takes? While dynamic composition is widely seen as an attractive feature of software services, for the present at least, its practical value appears to be 'unproven'.

9.6 Interpreting the Outcomes: The Experiments

In the thesis, we perform two studies, a close replication and extension study which are presented in Chapters 7 and 8 respectively. In each study, we conduct an experiment for assessment purpose. The following paragraphs provide an interpretation of the results from each study.

The first experiment is conducted to validate our close replication model. The validation has been done by comparison between the results of this simulation with the results of the original study. The purpose of the experiment is to evaluate the time required for generation of composition plans. The generation time consists of three execution times which we investigate. The first execution time involves checking the mode, binding and operation semantic composability. The second execution time involves checking the message composability. The last execution time related to checking the composition soundness. The result shows that these three execution times are similar in terms of profile with the original study. Therefore, it is considered as demonstrating the validity of our model.

The purpose of the extension study is to investigate the use of a QoS model as the selection mechanism, replacing the ranking mechanism used in the original study. We perform an experiment to compare the performance of using QoS model selection mechanism with the ranking selection mechanism. The results show that the overall QoS of selected composition plans has highest quality score

when using QoS model. However, for individual qualities, there are few cases where the ranking mechanism produces better values. However, the number of these individual qualities is always less than half of the qualities that are used in the comparison.

9.7 Summary

This chapter discussed the work that has been done throughout this thesis. We discussed the SLR and the experiments in terms of threat to validity. For the threat to validity, our discussion focused on the internal validity for both. For the SLR, we do not aim to generalise the outcomes of the SLR beyond the specific topic of service composition. As far as the experiments are concerned, the conditions are sufficiently controlled (for the close replication study, and as well for the extension study). We discussed the classification issues related to the SLR and its outcomes. Finally, we interpreted the outcomes for both the SLR and the experiments.

Chapter 10

Conclusions

10.1 Introduction

This thesis describes an approach intended to enrich the scientific knowledge about dynamic Web service composition. The study set out to investigate the dynamic Web service composition paradigm. The investigation has been performed through two elements, conducting a mapping study and developing a simulation framework. Firstly, the mapping study has been used to explore the strategies and approaches that have been used for dynamically composing atomic services to form composite services and for which empirical outcomes are available. Secondly, the thesis uses a simulation framework to experimentally study two strategies for Web service composition. These are Semantic Web and QoS model strategies for service composition, which have been studied in more detail through the use of a close replication and an extension study.

The contributions presented in this thesis may influence the research debate on some aspects (such as using simulation as investigation tool, using Semantic Web, QoS models) with respect to Web service composition. Accordingly, in this chapter we review how well we are able to answer our initial questions, and what approaches might be appropriate for improving these answers. Also we identify some possible areas for future work.

10.2 The Simulation Framework

The major aspect of this study is the development of a simulation framework to study dynamic Web service composition. We developed our simulation model with regard to the specific research questions that we asked.

- *how simulation can be used to study Web service composition issues?*
- *Can simulation be used as a research method to replicate a study from Semantic Web strategy?*
- If the outcomes from the replication study proof the validity of our simulation, then, *Can simulation model be used for further investigation into the service composition process to compare a QoS model and ranking mechanism for service selection?*

As mentioned earlier in Chapter 1, the simulation framework is particularly used to study semantic Web and QoS model strategies for Web service composition that have been identified in our mapping study. These two strategies are studied through both a close replication study, which focuses on the matchmaking process, and an extension study, which focuseson the selection process. The main empirical findings of these studies are summarised within the respective empirical chapters: A Simulation to Replicate a Web Service Composition Study and Web service selection. The following subsections seek to synthesise the empirical findings of these studies.

10.2.1 Empirical Findings: A Close Replication Study

The close replication study (Chapter 7) shows how the simulation model has been built upon a 'baseline' existing study that is identified through the evaluation of the outcomes from the mapping study. The simulation model is implemented and used to replicate the 'baseline' study. The outcomes of the experiments have similar profiles to the outcomes from the baseline study. We interpret the outcomes of the experiments as follow:

1. The similarity between the outcomes from both studies are considered as a proof of validity of our simulation model. Accordingly, we build up confidence in using our simulation model for further investigation in Web service composition. In addition, the simulation help with trying and testing different ideas in order to improve our model, thereby, improve our approach for Web service composition.

2. Demonstrating the replicability of the 'baseline' outcomes also gives credibility for its results. The replication study provides another attempt to test a set of composabilty rules presented by the algorithm used in the 'baseline' study to compare syntactic and semantic characteristics of Web services. The experiment results confirm that checking the message composability between two operations takes most of the time of the service composition process.

3. Although the literature shows the lack of real data-sets that have been described by using Semantic Web, the simulation can help with generating data-sets that have appropriate semantic characteristics in order to be used for exploring our model.

10.2.2 Empirical Findings: Web Service Selection Study

Chapter 8 presents an extension study that is derived from the close replication study. The extension study employes a QoS model for the Web service selection process. The following list gives details about the contribution of this study

1. The study presents a general and extensible QoS model which consists of generic quality criteria that all Web services should adopt. We extend the definitions that are used in the 'baseline' study, and which are used to semantically describe the functional features of the Web service. The model used for the extension has been employed to describe the quality criteria (non-functional features) of the Web service, which also can be used to add more quality criteria to the QoS model such as domain specific quality criteria. The QoS model is used to describe the quality criteria of Web service component and as well for the composite service.

2. Based on the presentation of the QoS model, we describe a service selection approach that uses Multi-attribute Utility Theory (MAUT) to compute optimal execution plans for composite services. The quality criteria of these optimal execution plans can be scaled and weighted to select a composition plan that has a higher quality score according to a user's priorities.

3. We describes how the QoS model and its computation have been implemented and integrated into the simulation framework.

4. We conduct experiments to compare the proposed QoS model with the ranking selection mechanism that has been used in the 'baseline' study. The results show that the QoS model can reliably select high quality execution plans (i.e., plans which have higher overall QoS). Consequently, the ranking selection mechanism can be replaced by the QoS model for better performance in terms of selecting a composition plan that has a high quality score value.

10.3 Future Work

Our motivation for this research was to identify where we could most usefully make a contribution to understanding the strengths and weaknesses of particular composition strategies. The service concept, and its realisation through Web services is still relatively immature. So it is not entirely surprising that we did not find a rich set of experiences from which to draw any conclusions.

To address this, we investigated the use of a simulation framework that would allow us to turn service composition into more of a knowledge problem, and hence to be able to make comparisons between different strategies, while accepting that the wide variation in these does impose some limitation upon the measures that we can use.

In the following list, we suggest some ongoing research issues that could usefully extend this work.

- The mapping study identified a number of strategies for Web service composition, where each strategy offers an interesting area to conduct a fuller

systematic literature review. Initially, we considered choosing one of the strategies (QoS) that has been studied in this thesis for further investigation.

- This thesis used Semantic Web to define the functional and non-functional (quality criteria) properties of Web services which facilitate the automation of the discovery, matchmaking and selection processes. It should be possible to extend the semantic definition to deal with other properties of the Web service, such as a precondition that have to be satisfied before interacting with a Web service. The precondition may determine requirements that must be satisfied, such as, only a consumer from the UK can be served, or a consumer must have an account with the service provider. The definition of the precondition of the Web service will take a form of a logical expression and must be satisfied in order to invoke the service.

- Chapter 8 studied the QoS model experimentally and in doing so it mainly focused on the generic quality criteria. Future work could investigate including other quality criteria that are related to specific domains in the QoS model. The investigation could study how each domain defines and computes their quality criteria.

- The simulation framework covered two processes of the generic life-cycle model for Web service composition. Ongoing research needs to address support for exception handling during composite service runtime. For example, if a WS involved in a composition plan becomes unavailable or fails during execution, this step specifies what action should be taken,such as finding another WS to replace the unavailable or failed one, or re-planning the whole process of the service composition.

Bibliography

Anissa Abrougui, Annabelle Mercier, Michel Occello, and Marc-Philippe Huget. Recursive multi-agent system for dynamic and adaptative web services composition. In *MEDES '09: Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, pages 295–299, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-829-2. doi: http://doi.acm.org/10.1145/1643823.1643877.

Evon M. O. Abu-Taieh and Asim Abdel Rahman. El-Sheikh. *Handbook of Research on Discrete Event Simulation Environments: Technologies and Applications*, chapter 1, pages 1–14. Information Science Reference (an imprint of IGI Global) , New York, 2010.

Rama Akkiraju, Joel Farrell, John Miller, Meenakshi Nagarajan, Marc-Thomas Schmidt, Amit Sheth, and Kunal Verma. Web service semantics - WSDL-S. W3C, November 2005. URL `http://www.w3.org/Submission/WSDL-S/`. Version 1.0.

Khaldoon Al-Zoubi and Gabriel Wainer. Using rest web-services architecture for distributed simulation. In *Proceedings of the 2009 ACM/IEEE/SCS 23rd Workshop on Principles of Advanced and Distributed Simulation*, PADS '09, pages 114–121, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3713-9. doi: http://dx.doi.org/10.1109/PADS.2009.16. URL `http://dx.doi.org/10.1109/PADS.2009.16`.

José Luis Ambite and Matthew Weathers. Automatic composition of aggregation workflows for transportation modeling. In *Proceedings of the 2005 national conference on Digital government research*, DG.O '05,

pages 41–49. Digital Government Society of North America, 2005. URL http://portal.acm.org/citation.cfm?id=1065226.1065240.

M.A. Aslam, Jun Shen, S. Auer, and M. Herrmann. An integration life cycle for semantic web services composition. In *Computer Supported Cooperative Work in Design, 2007. CSCWD 2007. 11th International Conference on*, pages 490 –495, april 2007. doi: 10.1109/CSCWD.2007.4281485.

Youakim Badr, Ajith Abraham, Frédérique Biennier, and Crina Grosan. Enhancing web service selection by user preferences of non-functional features. In *Proceedings of the 2008 4th International Conference on Next Generation Web Services Practices*, NWESP '08, pages 60–65, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3455-8. doi: 10.1109/NWeSP.2008.39. URL http://dx.doi.org/10.1109/NWeSP.2008.39.

Jerry Banks, John Carson, Barry L Nelson, and David Nicol. *Discrete-Event System Simulation, Fourth Edition*. Prentice Hall, 2004.

Luciano Baresi, Carlo Ghezzi, and Sam Guinea. Smart monitors for composed services. In *Proceedings of the 2nd international conference on Service oriented computing*, ICSOC '04, pages 193–202, New York, NY, USA, 2004. ACM. ISBN 1-58113-871-7. doi: http://doi.acm.org/10.1145/1035167.1035195. URL http://doi.acm.org/10.1145/1035167.1035195.

Orriëns Bart, Yang Jian, and Papazoglou Mike P. Servicecom: A tool for service composition reuse and specialization. In *Proceedings of Web Information Systems Engineering*, pages 355–358. IEEE Computer Society, 2003.

R.R. Barton. Simulation experiment design. In *Simulation Conference (WSC), Proceedings of the 2010 Winter*, pages 75 –86, dec. 2010. doi: 10.1109/WSC.2010.5679171.

Khalid Belhajjame, Suzanne M. Embury, Norman W. Paton, Robert Stevens, and Carole A. Goble. Automatic annotation of web services based on workflow definitions. *ACM Trans. Web*, 2(2):1–34, 2008. ISSN 1559-1131. doi: http://doi.acm.org/10.1145/1346237.1346239.

Liu Bing and Chen Huaping. Web service composition and analysis: A petri-net based approach. In *Semantics, Knowledge and Grid, 2005. SKG '05. First International Conference on*, pages 111 –111, nov. 2005. doi: 10.1109/SKG.2005.143.

M. Brian Blake and Hassan Gomaa. Agent-oriented compositional approaches to services-based cross-organizational workflow. *Decision Support Systems*, 40(1):31–50, 2005.

J. J. Bolinger, P. Ghose, J. H. Sosinski, and W. F. Esser. Decision analysis utilizing multi-attribute utility theory in engineering evaluations. *Power Apparatus and Systems, IEEE Transactions on*, PAS-97(4):1245 –1253, july 1978. ISSN 0018-9510. doi: 10.1109/TPAS.1978.354607.

Benatallah Boualem, Dumas Marlon, and Sheng Quan Z. Facilitating the rapid development and scalable orchestration of composite web services. *Distributed and Parallel Databases*, 17(1):5–37, 2005.

O.Pearl Brereton, David Budgen, K Bennett, M Munro, P Layzell, L Macaulay, D Griffiths, and C Stannett. The Future of Software. *Communications of the ACM*, 42(12):78–84, December 1999.

O.Pearl Brereton, N.E. Gold, D. Budgen, K.H. Bennett, and N.D. Mehandjiev. Systematic literature review: a pilot study of service-based systems. Technical report, Keele University, 2006.

O.Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *J. Syst. Softw.*, 80(4):571–583, 2007. ISSN 0164-1212.

A. Brooks, J. Daly, J. Miller, M. Roper, and M. Wood. Replication's role in experimental computer science, 1994.

Jian Cao, Jie Wang, Shensheng Zhang, and Minglu Li. A dynamically reconfigurable system based on workflow and service agents. *Eng. Appl. Artif. Intell.*, 17(7):771–782, 2004. ISSN 0952-1976. doi: http://dx.doi.org/10.1016/j.engappai.2004.08.030.

Massimo Paolucci Carnegie and Massimo Paolucci. Delivering semantic web services, 2003.

Fabio Casati, Ski Ilnicki, Li-jie Jin, Vasudev Krishnamoorthy, and Ming-Chien Shan. Adaptive and dynamic service composition in eFlow. In *CAiSE '00: Proceedings of the 12th International Conference on Advanced Information Systems Engineering*, pages 13–31. Springer-Verlag, 2000. ISBN 3-540-67630-9.

Sodki Chaari, Youakim Badr, and Frédérique Biennier. Enhancing web service selection by qos-based ontology and ws-policy. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 2426–2431, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-753-7. doi: http://doi.acm.org/10.1145/1363686.1364260.

Girish B. Chafle, Sunil Chandra, Vijay Mann, and Mangala Gowri Nanda. Decentralized orchestration of composite web services. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, WWW Alt. '04, pages 134–143, New York, NY, USA, 2004. ACM. ISBN 1-58113-912-8. doi: http://doi.acm.org/10.1145/1013367.1013390. URL `http://doi.acm.org/10.1145/1013367.1013390`.

D. Chakraborty, Y. Yesha, and A. Joshi. A distributed service composition protocol for pervasive environments. In *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, volume 4, pages 2575 – 2580 Vol.4, 2004. doi: 10.1109/WCNC.2004.1311494.

Senthilanand Chandrasekaran, Gregory Silver, John A. Miller, Jorge Cardoso, and Amit P. Sheth. Web service technologies and their synergy with simulation. In *Proceedings os the 34th conference on Winter simulation: exploring new frontiers*, pages 606–615. Winter Simulation Conference, 2002.

Wei-Lun Chang. Using multi-attribute utility theory to rank and select co-branding partners. In *Intelligent Systems Design and Applications, 2008. ISDA '08. Eighth International Conference on*, volume 1, pages 195 –200, nov. 2008. doi: 10.1109/ISDA.2008.220.

Wei-Lun Chang and Chia-Yun Ho. A mixed-initiative model for quality-based e-services pricing. In *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pages 574 –579, april 2010. doi: 10.1109/ITNG.2010.169.

Ching-Wen Chen and Chuan-Chi Weng. A power efficiency routing and maintenance protocol in wireless multi-hop networks. *J. Syst. Softw.*, 85(1):62–76, January 2012. ISSN 0164-1212. doi: 10.1016/j.jss.2011.07.012. URL `http://dx.doi.org/10.1016/j.jss.2011.07.012`.

Zhumin Chen, Jun Ma, Ling Song, and Li Lian. An efficient approach to web services discovery and composition when large scale services are available. In *Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing*, pages 34–41, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2751-5. doi: 10.1109/APSCC.2006.29. URL `http://portal.acm.org/citation.cfm?id=1191822.1191996`.

W.K. Cheung, Jiming Liu, Kevin H. Tsang, and R.K. Wong. Dynamic resource selection for service composition in the grid. In *Web Intelligence, 2004. WI 2004. Proceedings. IEEE/WIC/ACM International Conference on*, pages 412 – 418, sept. 2004. doi: 10.1109/WI.2004.10080.

Mohan Baruwal Chhetri, Ingo Mueller, Suk Keong Goh, and Ryszard Kowalczyk. Asapm - an agent-based framework for adaptive management of composite service lifecycle. In *Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, WI-IATW '07, pages 444–448, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-3028-1. URL `http://dl.acm.org/citation.cfm?id=1339264.1339744`.

Yu-Liang Chi and Hsun-Ming Lee. A formal modeling platform for composing web services. *Expert Syst. Appl.*, 34:1500–1507, February 2008. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2007.01.022. URL `http://dx.doi.org/10.1016/j.eswa.2007.01.022`.

Gao Chunming, Liu Rongsheng, Song Yan, and Chen Huowang. A model checking tool embedded into services composition environment. In *Proceeding*

of Grid and Cooperative Computing, pages 355–362. IEEE Computer Society Press, 2006.

Asit Dan, Heiko Ludwig, and Giovanni Pacifici. Web service differentiation with service level agreements. http://www.ibm.com/developerworks/library/ws-slafram/, May 2003. IBM, Software Group.

Massimiliano Di Penta, Raffaele Esposito, Maria Luisa Villani, Roberto Codato, Massimiliano Colombo, and Elisabetta Di Nitto. Ws binder: a framework to enable dynamic binding of composite web services. In *SOSE '06: Proceedings of the 2006 international workshop on Service-oriented software engineering*, pages 74–80, New York, NY, USA, 2006. ACM. ISBN 1-59593-398-0. doi: http://doi.acm.org/10.1145/1138486.1138502.

Wen-Li Dong, Hang Yu, and Yu-Bing Zhang. Testing bpel-based web service composition using high-level petri nets. In *Enterprise Distributed Object Computing Conference, 2006. EDOC '06. 10th IEEE International*, pages 441 –444, 2006. doi: 10.1109/EDOC.2006.59.

Zhihui Du, Jingkun Hu, Yinong Chen, Zhili Cheng, and Xiaoying Wang. Controversy corner: Optimized qos-aware replica placement heuristics and applications in astronomy data grid. *J. Syst. Softw.*, 84(7):1224–1232, July 2011. ISSN 0164-1212. doi: 10.1016/j.jss.2011.02.038. URL `http://dx.doi.org/10.1016/j.jss.2011.02.038`.

Christophe Dumez, Ahmed Nait-sidi moh, Jaafar Gaber, and Maxime Wack. Modeling and specification of web services composition using UML-S. In *NWESP '08: Proceedings of the 2008 4th International Conference on Next Generation Web Services Practices*, pages 15–20, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3455-8. doi: http://dx.doi.org/10.1109/NWeSP.2008.17.

Schahram Dustdar and Wolfgang Schreiner. A survey on web services composition. *Int. J. Web Grid Serv.*, 1(1):1–30, 2005. ISSN 1741-1106. doi: http://dx.doi.org/10.1504/IJWGS.2005.007545.

Thomas Erl. *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004. ISBN 0131428985.

Norman E. Fenton and Shari Lawrence Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Co., Boston, MA, USA, 2nd edition, 1998. ISBN 0534954251.

Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. In *Proceedings of the 22nd international conference on Software engineering*, ICSE '00, pages 407–416, New York, NY, USA, 2000. ACM. ISBN 1-58113-206-9. doi: http://doi.acm.org/10.1145/337180.337228. URL `http://doi.acm.org/10.1145/337180.337228`.

Daniela Florescu, Andreas Grünhagen, and Donald Kossmann. Xl: an xml programming language for web service specification and composition. *Computer Networks*, 42(5):641 – 660, 2003. ISSN 1389-1286. doi: 10.1016/S1389-1286(03)00224-X. URL `http://www.sciencedirect.com/science/article/pii/S138912860300224X`.

Breno França and Guilherme Travassos. Reporting guidelines for simulation-based studies in software engineering. In *proceedings of the EASE 2012*. IET, 2012.

Yujian Fu, Zhijiang Dong, and Xudong He. Modeling, validating and automating composition of web services. In *Proceedings of the 6th international conference on Web engineering*, ICWE '06, pages 217–224, New York, NY, USA, 2006. ACM. ISBN 1-59593-352-2. doi: http://doi.acm.org/10.1145/1145581.1145626. URL `http://doi.acm.org/10.1145/1145581.1145626`.

Keita Fujii and Tatsuya Suda. Component service model with semantics (CoSMoS): A new component model for dynamic service composition. In *Proceedings of Applications and the Internet Workshops*, pages 348–354. IEEE Computer Society Press, 2004a.

Keita Fujii and Tatsuya Suda. Dynamic service composition using semantic information. In *ICSOC '04: Proceedings of the 2nd international conference on*

Service oriented computing, pages 39–48, New York, NY, USA, 2004b. ACM. ISBN 1-58113-871-7. doi: http://doi.acm.org/10.1145/1035167.1035174.

Keita Fujii and Tatsuya Suda. Semantics-based dynamic service composition. *Selected Areas in Communications, IEEE Journal on*, 23(12):2361 – 2372, dec. 2005. ISSN 0733-8716. doi: 10.1109/JSAC.2005.857202.

Jeannine Hall Gailey. *Understanding Web Services Specifications and the WSE*. Microsoft Press, 1st edition edition, February 2004.

John Garofalakis, Yannis Panagis, Evangelos Sakkopoulos, and Athanasios Tsakalidis. Web service discovery mechanisms: Looking for a needle in a haystack? In *In: International Workshop on Web Engineering. (2004*, 2004.

Kristof Geebelen, Sam Michiels, and Wouter Joosen. Dynamic reconfiguration using template based web service composition. In *MW4SOC '08: Proceedings of the 3rd workshop on Middleware for service oriented computing*, pages 49–54, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-368-6. doi: http://doi.acm.org/10.1145/1462802.1462811.

R.L. Glass, V. Ramesh, and I. Vessey. An Analysis of Research in Computing Disciplines. *Communications of the ACM*, 47:89–94, June 2004.

Jeffrey Gortmaker, Marijn Janssen, and René W. Wagenaar. The advantages of web service orchestration in perspective. In *Proceedings of the 6th international conference on Electronic commerce*, ICEC '04, pages 506–515, New York, NY, USA, 2004. ACM. ISBN 1-58113-930-6. doi: http://doi.acm.org/10.1145/1052220.1052284. URL `http://doi.acm.org/10.1145/1052220.1052284`.

Xiaohui Gu, Klara Nahrstedt, Rong N. Chang, and Christopher Ward. QoS-assured service composition in managed service overlay networks. In *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 194, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1920-2.

Xiaohui Gu, Klara Nahrstedt, and Bin Yu. Spidernet: An integrated peer-to-peer service composition framework. In *Proceedings of the 13th*

IEEE International Symposium on High Performance Distributed Computing, pages 110–119, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7803-2175-4. doi: 10.1109/HPDC.2004.32. URL http://portal.acm.org/citation.cfm?id=1032647.1033297.

Richard S. Hall and Humberto Cervantes. Gravity: supporting dynamically available services in client-side applications. volume 28, pages 379–382, New York, NY, USA, September 2003. ACM. doi: http://doi.acm.org/10.1145/949952.940126. URL http://doi.acm.org/10.1145/949952.940126.

Rachid Hamadi and Boualem Benatallah. A petri net-based model for web service composition. In *Proceedings of the 14th Australasian database conference - Volume 17*, ADC '03, pages 191–200, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc. ISBN 0-909-92595-X. URL http://portal.acm.org/citation.cfm?id=820085.820121.

Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. ISBN 1558609016.

Yanan Hao and Yanchun Zhang. Web services discovery based on schema matching. In *Proceedings of the thirtieth Australasian conference on Computer science - Volume 62*, ACSC '07, pages 107–113, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc. ISBN 1-920-68243-0. URL http://dl.acm.org/citation.cfm?id=1273749.1273762.

Chuan He, Xiaomin Zhu, Hui Guo, Dishan Qiu, and Jianqing Jiang. Rolling-horizon scheduling for energy constrained distributed real-time embedded systems. *Journal of Systems and Software*, 85(4):780 – 794, 2012. ISSN 0164-1212. doi: 10.1016/j.jss.2011.10.008. URL http://www.sciencedirect.com/science/article/pii/S0164121211002615.

Liu Hong-yu and Li Ying. Coordinative development of urban traffic and transportation system using multi-attribute utility theory. In *Environmental Science and Information Application Technology (ESIAT), 2010 International Conference on*, volume 2, pages 326 –329, july 2010. doi: 10.1109/ESIAT.2010.5567362.

Gang Huang, Weihu Wang, Tiancheng Liu, and Hong Mei. Simulation-based analysis of middleware service impact on system reliability: Experiment on java application server. *J. Syst. Softw.*, 84(7):1160–1170, July 2011. ISSN 0164-1212. doi: 10.1016/j.jss.2011.02.008. URL `http://dx.doi.org/10.1016/j.jss.2011.02.008`.

Microsoft SAP AG Siebel Systems IBM, BEA Systems. Business process execution language for web services version 1.1. http://www.ibm.com/developerworks/library/specification/ws-bpel/, Feb. 2007.

Nirmal Mukhi IBM. Business process with bpel4ws: Learning bpel4ws, part 4. http://www.ibm.com/developerworks/webservices/library/ws-bpelcol4/, Nov. 2002.

Noha Ibrahim and Frédéric Le Mouël. A survey on service composition middleware in pervasive environments. *International Journal of Computer Science Issues*, 1:1–12, 2009.

Ko In-Young, Yao Ke-Thia, and Neches Robert. Dynamic coordination of information management services for processing dynamic web content. In *Proceedings of the 11th international conference on World Wide Web*, pages 355–365. ACM Press, 2002.

Kimihito Ito and Yuzuru Tanaka. A visual environment for dynamic web application composition. In *HYPERTEXT '03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 184–193, New York, NY, USA, 2003. ACM. ISBN 1-58113-704-4. doi: http://doi.acm.org/10.1145/900051.900092.

G.T. Jayaputera, A. Zaslavsky, and S.W. Loke. Enabling run-time composition and support for heterogeneous pervasive multi-agent systems. *Journal of Systems and Software*, 80(12):2039 – 2062, 2007. ISSN 0164-1212. doi: DOI: 10.1016/j.jss.2007.03.013. URL `http://www.sciencedirect.com/science/article/B6V0N-4N9DJS8-1/2/5bedfba425a0-6b44eafc69dc22905312`. Selected papers from the International Conference on Pervasive Services (ICPS 2006).

A. Jedlitschka and D. Pfahl. Reporting guidelines for controlled experiments in software engineering. In *Empirical Software Engineering, 2005. 2005 International Symposium on*, page 10 pp., nov. 2005. doi: 10.1109/ISESE.2005.1541818.

Zhang Jianhong, Zhang Shensheng, Cao Jian, and Mou Yujie. Improved HTN planning approach for service composition. In *Services Computing, 2004. (SCC 2004). Proceedings. 2004 IEEE International Conference on*, pages 609 – 612, sept. 2004. doi: 10.1109/SCC.2004.1358075.

Kim Jihie, Spraragen Marc, and Gil Yolanda. An intelligent assistant for interactive workflow composition. In *Proceedings of the 9th international conference on Intelligent user interfaces IUI'04*, pages 125–131. ACM Press, 2004.

Nakazawa Jin and Tokuda Hideyuki. A pluggable service-to-service communication mechanism for home multimedia networks. In *Proceedings of the Tenth ACM international Conference on Multimedia*, pages 612–630. ACM Press, 2002.

Yan Jun, Kowalczyk Ryszard, Lin Jian, Chhetri Mohan B., Goh Suk Keong, and Zhang Jianying. Autonomous service level agreement negotiation for service composition provision. *Future Generation Computer Systems*, 23(6): 748–759, 2007.

Lee Jun-Yi, Hsiao Hung-Chang, King Chung-Ta, and Banerjee Amit. Canal: A distributed service composition system using mobile agents. In *Proceedings of Information Technology: Research and Education*, pages 449–453. IEEE Computer Society Press, 2005.

Natalia Juristo and Sira Vegas. Using differences among replications of software engineering experiments to gain knowledge. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, ESEM '09, pages 356–366, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-1-4244-4842-5. doi: http://dx.doi.org/10.1109/ESEM.2009.5314236. URL `http://dx.doi.org/10.1109/ESEM.2009.5314236`.

Swaroop Kalasapur, Mohan Kumar, and Behrooz Shirazi. Evaluating service oriented architectures (SOA) in pervasive computing. In *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications*, pages 276–285, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2518-0. doi: 10.1109/PERCOM.2006.24. URL `http://portal.acm.org/citation.cfm?id=1128015.1128351`.

Sangseung Kang and Joochan Sohn. A service template authoring environment for urc service composition. In *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, volume 3, pages 4–1572, 2006. doi: 10.1109/ICACT.2006.206285.

Georgia M. Kapitsaki, Dimitrios A. Kateros, Christos A. Pappas, Nikolaos D. Tselikas, and Iakovos S. Venieris. Model-driven development of composite web applications. In *iiWAS '08: Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, pages 399–402, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-349-5. doi: http://doi.acm.org/10.1145/1497308.1497380.

Mick Kerrigan. Web service selection mechanisms in the web service execution environment (wsmx). In *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, pages 1664–1668, New York, NY, USA, 2006. ACM. ISBN 1-59593-108-2. doi: http://doi.acm.org/10.1145/1141277.1141671. URL `http://doi.acm.org/10.1145/1141277.1141671`.

Barbara A. Kitchenham. The role of replications in empirical software engineering–a word of warning. *Empirical Softw. Engg.*, 13:219–221, April 2008. ISSN 1382-3256. doi: 10.1007/s10664-008-9061-0. URL `http://dl.acm.org/citation.cfm?id=1361580.1361584`.

Barbara A. Kitchenham, Ieee Computer Society, Shari Lawrence Pfleeger, Lesley M. Pickard, Peter W. Jones, David C. Hoaglin, Khaled El Emam, and Ieee Computer Society. Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, 28:721–734, 2002.

Barbara A. Kitchenham, S. Charters, , David Budgen, Pearl Brereton, Mark Turner, Steve Linkman, Magne Jorgensen, Emilia Mendes, and Giuseppe Visaggio. Guidelines for performing systematic literature reviews in software engineering. Technical report, Keele University and Durham University, 2007.

Barbara A. Kitchenham, David Budgen, and O. Pearl Brereton. Using mapping studies as the basis for further research—a participant-observer case study. *Information & Software Technology*, 53(4):638–651, 2011. Special section from EASE 2010.

Masakatsu Kosuga, Naoki Kirimoto, Tatsuya Yamazaki, Tomonori Nakanishi, Masakazu Masuzaki, and Kazuo Hasuike. A multimedia service composition scheme for ubiquitous networks. *J. Netw. Comput. Appl.*, 25:279–293, October 2002. ISSN 1084-8045. doi: 10.1006/jnca.2002.0140. URL `http://portal.acm.org/citation.cfm?id=637588.637592`.

Averill M Law. *Simulation Modeling and Analysis, Fourth Edition.* TATA McGraw Hill, 2008.

A. Lazcano, G. Alonso, C. Schuler, and C. Schuler. The wise approach to electronic commerce. *International Journal of Computer Systems Science & Engineering, special issue on Flexible Workflow Technology Driving the Networked Economy*, 15:2000, 2000.

Ljubomir Lazić and Nikos Mastorakis. Applying modeling & simulation to the software testing process: one test oracle solution. In *Proceedings of the 7th WSEAS international conference on Automatic control, modeling and simulation*, ACMOS'05, pages 248–256, Stevens Point, Wisconsin, USA, 2005. World Scientific and Engineering Academy and Society (WSEAS). ISBN 960-8457-12-2. URL `http://portal.acm.org/citation.cfm?id=1983314.1983364`.

Frank Leymann. Web services flow language (WSFL 1.0), March 2008. URL `http://xml.coverpages.org/WSFL-Guide-200110.pdf`.

Wen-Yau Liang, Chun-Che Huang, and Horng-Fu Chuang. The design with object (DwO) approach to web services composition. *Comput. Stand. Interfaces*,

29:54–68, January 2007. ISSN 0920-5489. doi: 10.1016/j.csi.2005.11.001. URL http://portal.acm.org/citation.cfm?id=1222223.1222366.

Noura Limam, Joanna Ziembicki, Reaz Ahmed, Youssef Iraqi, Dennis Tianshu Li, Raouf Boutaba, and Fernando Cuervo. OSDA: Open service discovery architecture for efficient cross-domain service provisioning. *Comput. Commun.*, 30:546–563, February 2007. ISSN 0140-3664. doi: 10.1016/j.comcom.2005.11.017. URL http://portal.acm.org/citation.cfm?id=1224247.1224421.

Jenn-Wei Lin and Shih-Chieh Tang. A grid-based coverage approach for target tracking in hybrid sensor networks. *J. Syst. Softw.*, 84(10):1746–1756, October 2011. ISSN 0164-1212. doi: 10.1016/j.jss.2011.05.010. URL http://dx.doi.org/10.1016/j.jss.2011.05.010.

R. Murray Lindsay and A. S. C. Ehrenberg. The design of replicated studies. *The American Statistician*, 4(3):217–228, August 1993. URL http://www.jstor.org/stable/2684982.

An Liu, Qing Li, Liusheng Huang, and Mingjun Xiao. Facts: A framework for fault-tolerant composition of transactional web services. *Services Computing, IEEE Transactions on*, 3(1):46 –59, jan. 2010. ISSN 1939-1374. doi: 10.1109/TSC.2009.28.

Huawen Liu and Shichao Zhang. Noisy data elimination using mutual k-nearest neighbor for classification mining. *J. Syst. Softw.*, 85(5):1067–1074, May 2012. ISSN 0164-1212. doi: 10.1016/j.jss.2011.12.019. URL http://dx.doi.org/10.1016/j.jss.2011.12.019.

Na Liu, John Grundy, and John Hosking. A visual language and environment for composing web services. In *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, ASE '05, pages 321–324, New York, NY, USA, 2005. ACM. ISBN 1-58113-993-4. doi: http://doi.acm.org/10.1145/1101908.1101960. URL http://doi.acm.org/10.1145/1101908.1101960.

Yin Liu, Wenyin Liu, and Changjun Jiang. Object-process diagrams as explicit graphic tool for web service composition. *J. Integr. Des.*

Process Sci., 8:113–127, January 2004a. ISSN 1092-0617. URL http://portal.acm.org/citation.cfm?id=1275864.1275872.

Yutu Liu, Anne H. Ngu, and Liang Z. Zeng. QoS computation and policing in dynamic web service selection. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 66–73, New York, NY, USA, 2004b. ACM. ISBN 1-58113-912-8. doi: http://doi.acm.org/10.1145/1013367.1013379.

Pedro Lopes, Joel Arrais, and José Luís Oliveira. Dynamic service integration using web-based workflows. In *iiWAS '08: Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, pages 622–625, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-349-5. doi: http://doi.acm.org/10.1145/1497308.1497426.

Qin Lu, Weishi Zhang, Bo Su, and Xiuguo Zhang. Exception handling policies for composite web services and their formal description. In *Network and Parallel Computing Workshops, 2007. NPC Workshops. IFIP International Conference on*, pages 793 –798, sept. 2007. doi: 10.1109/NPC.2007.13.

Zheng Lu, Peter Hyland, Aditya K. Ghose, and Ying Guan. Using assumptions in service composition context. In *Proceedings of the 2006 international workshop on Service-oriented software engineering*, SOSE '06, pages 19–25, New York, NY, USA, 2006. ACM. ISBN 1-59593-398-0. doi: http://doi.acm.org/10.1145/1138486.1138491. URL http://doi.acm.org/10.1145/1138486.1138491.

Zakaria Maamar, Soraya Kouadri, and Hamdi Yahyaoui. A web services composition approach based on software agents and context. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 1619–1623, New York, NY, USA, 2004. ACM. ISBN 1-58113-812-1.

Therani Madhusudan and Young-Jun Son. A simulation-based approach for dynamic process management at web service platforms. *Computers and Industrial Engineering*, 49(2):287–317, 2005.

Therani Madhusudan and N. Uttamsingh. A declarative approach to composing web services in dynamic environments. *Decis. Support Syst.*, 41(2):325–357, 2006. ISSN 0167-9236. doi: http://dx.doi.org/10.1016/j.dss.2004.07.003.

Mika V. Mäntylä, Casper Lassenius, and Jari Vanhanen. Rethinking replication in software engineering: Can we see the forest for the trees?, May 2010.

Braem Mathieu, Joncheere Niels, Vanderperren Wim, Straeten Ragnhild Van Der, and Jonckers Viviane. Guiding service composition in a visual service creation environment. In *Proceedings of the European Conference on Web Services ECOWS*, pages 13–22. IEEE Computer Society Press, 2006.

Brahim Medjahed. Semantic web enabled composition of web services. Master's thesis, Virginia Polytechnic Institute and State University, 2004.

Brahim Medjahed, Athman Bouguettaya, and Ahmed K. Elmagarmid. Composing web services on the semantic web. *The VLDB Journal*, 12:333–351, November 2003. ISSN 1066-8888. doi: http://dx.doi.org/10.1007/s00778-003-0101-5. URL `http://dx.doi.org/10.1007/s00778-003-0101-5`.

Sun Meng and Farhad Arbab. Web services choreography and orchestration in reo and constraint automata. In *Proceedings of the 2007 ACM symposium on Applied computing*, SAC '07, pages 346–353, New York, NY, USA, 2007. ACM. ISBN 1-59593-480-4. doi: http://doi.acm.org/10.1145/1244002.1244085. URL `http://doi.acm.org/10.1145/1244002.1244085`.

Microsoft. Web services for business process design(xlang), June 2001. URL `http://xml.coverpages.org/xlang.html`.

N. Milanovic and M. Malek. Current solutions for web service composition. *Internet Computing, IEEE*, 8(6):51 – 59, nov. 2004. ISSN 1089-7801. doi: 10.1109/MIC.2004.58.

James Miller. Replicating software engineering experiments: a poisoned chalice or the holy grail. *Inf. Softw. Technol.*, 47:233–244, March 2005. ISSN 0950-5849. doi: http://dx.doi.org/10.1016/j.infsof.2004.08.005. URL `http://dx.doi.org/10.1016/j.infsof.2004.08.005`.

Mahmoud Moadeli and Wim Vanderbauwhede. An analytical model of broadcast in qos-aware wormhole-routed nocs. *J. Syst. Softw.*, 84(1):12–20, January 2011. ISSN 0164-1212. doi: 10.1016/j.jss.2010.08.024. URL `http://dx.doi.org/10.1016/j.jss.2010.08.024`.

Sonia Ben Mokhtar, Nikolaos Georgantas, and Valérie Issarny. COCOA: Conversation-based service composition in pervasive computing environments with QoS support. *Systems and Software*, 80(12):1941–1955, 2007.

Istvan Molnar. *Handbook of Research on Discrete Event Simulation Environments: Technologies and Applications*, chapter Simulation: Body of Knowledge, pages 1–14. Information Science Reference, 2010.

Mangala Gowri Nanda, Satish Chandra, and Vivek Sarkar. Decentralizing execution of composite web services. In *Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, OOPSLA '04, pages 170–187, New York, NY, USA, 2004. ACM. ISBN 1-58113-831-8. doi: http://doi.acm.org/10.1145/1028976.1028991. URL `http://doi.acm.org/10.1145/1028976.1028991`.

Srini Narayanan and Sheila McIlraith. Analysis and simulation of web services. *Comput. Netw.*, 42:675–693, August 2003. ISSN 1389-1286. doi: 10.1016/S1389-1286(03)00228-7. URL `http://portal.acm.org/citation.cfm?id=873828.873836`.

Srini Narayanan and Sheila A. McIlraith. Simulation, verification and automated composition of web services. pages 77–88, 2002.

Eric Newcomer. *Understanding Web Services: XML, WSDL, SOAP and UDDI.* Addison Wesley, 2002.

Tiezheng Nie, Ge Yu, Derong Shen, Yue Kou, and Jie Song. An approach for composing web services on demand. In *Computer Supported Cooperative Work in Design, 2006. CSCWD '06. 10th International Conference on*, pages 1–6, may 2006. doi: 10.1109/CSCWD.2006.253170.

OASIS. Oasis uddi specification tc. Accessed 21-Oct-2010, Oct. 2010. URL `http://www.oasis-open.org/committees/uddi-spec/faq.php`.

Boon-Yaik Ooi, Huah-Yong Chan, and Yu-N. Cheah. Dynamic service placement and replication framework to enhance service availability using team formation algorithm. *Journal of Systems and Software*, (0):–, 2012. ISSN 0164-1212. doi: 10.1016/j.jss.2012.02.010. URL http://www.sciencedirect.com/science/article/pii/S0164121212000428?v=s5.

Seunghun Park and Doo-Hwan Bae. An approach to analyzing the software process change impact using process slicing and simulation. *J. Syst. Softw.*, 84(4):528–543, April 2011. ISSN 0164-1212. doi: 10.1016/j.jss.2010.11.919. URL http://dx.doi.org/10.1016/j.jss.2010.11.919.

Abhijit A. Patil, Swapna A. Oundhakar, Amit P. Sheth, and Kunal Verma. Meteor-s web service annotation framework. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 553–562, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X. doi: http://doi.acm.org/10.1145/988672.988747. URL http://doi.acm.org/10.1145/988672.988747.

Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs. "big"' web services: making the right architectural decision. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 805–814, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-085-2. doi: http://doi.acm.org/10.1145/1367497.1367606. URL http://doi.acm.org/10.1145/1367497.1367606.

Chris Peltz. Web services orchestration and choreography. *Computer*, 36(10): 46 – 52, oct. 2003. ISSN 0018-9162. doi: 10.1109/MC.2003.1236471.

M. Petticrew and H. Roberts. *Systematic reviews in the social sciences: A practical guide*. Malden, MA: Blackwell, 2006.

Dietmar Pfahl. Software process simulation frameworks in support of packaging and transferring empirical evidence. In *Proceedings of the 2006 international conference on Empirical software engineering issues: critical assessment and future directions*, pages 133–133, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-71300-5. URL http://portal.acm.org/citation.cfm?id=1767399.1767456.

L. Pickard, Barbara A. Kitchenham, and S.J. Linkman. Using simulated data sets to compare data analysis techniques used for software cost modelling. *Software, IEE Proceedings* -, 148(6):165 –174, dec 2001. ISSN 1462-5970. doi: 10.1049/ip-sen:20010621.

Shankar R. Ponnekanti and Armando Fox. Interoperability among independently evolving web services. In *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, Middleware '04, pages 331–351, New York, NY, USA, 2004. Springer-Verlag New York, Inc. ISBN 3-540-23428-4. URL `http://portal.acm.org/citation.cfm?id=1045658.1045682`.

Karl R. Popper. *The Logic of Scientific Discovery*. Hutchinson, London, 1959.

Stephen Potts and Mike Kopack. *Sams teach yourself web services in 24 hours*. Sams, Indianapolis, IN, USA, 2003. ISBN 9780768683721.

Hossein Pourreza and Peter Graham. On the fly service composition for local interaction environments. In *Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops*, PERCOMW '06, pages 393–, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2520-2. doi: http://dx.doi.org/10.1109/PERCOMW.2006.104. URL `http://dx.doi.org/10.1109/PERCOMW.2006.104`.

Lirong Qiu, Fen Lin, Changlin Wan, and Zhongzhi Shi. Semantic web services composition using ai planning of description logics. In *Services Computing, 2006. APSCC '06. IEEE Asia-Pacific Conference on*, pages 340 –347, 2006. doi: 10.1109/APSCC.2006.92.

Jinghai Rao, Peep Küngas, and Mihhail Matskin. Composition of semantic web services using linear logic theorem proving. *Inf. Syst.*, 31: 340–360, June 2006. ISSN 0306-4379. doi: 10.1016/j.is.2005.02.005. URL `http://portal.acm.org/citation.cfm?id=1140593.1140600`.

Irum Rauf, Anna Ruokonen, Tarja Systa, and Ivan Porres. Modeling a composite restful web service with uml. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA

'10, pages 253–260, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0179-4. doi: http://doi.acm.org/10.1145/1842752.1842801. URL http://doi.acm.org/10.1145/1842752.1842801.

Erik T. Ray. *Learning XML*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2003. ISBN 0596004206.

Casey Reardon, Eric Grobelny, Alan D. George, and Gongyu Wang. A simulation framework for rapid analysis of reconfigurable computing systems. *ACM Trans. Reconfigurable Technol. Syst.*, 3:25:1–25:29, November 2010. ISSN 1936-7406. doi: http://doi.acm.org/10.1145/1862648.1862655. URL http://doi.acm.org/10.1145/1862648.1862655.

Abdelmounaam Rezgui, Mourad Ouzzani, Athman Bouguettaya, and Brahim Medjahed. Preserving privacy in web services. In *Proceedings of the 4th international workshop on Web information and data management*, WIDM '02, pages 56–62, New York, NY, USA, 2002. ACM. ISBN 1-58113-593-9. doi: 10.1145/584931.584944. URL http://doi.acm.org/10.1145/584931.584944.

Jon Robinson, Ian Wakeman, and Tim Owen. Scooby: middleware for service composition in pervasive computing. In *Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, MPAC '04, pages 161–166, New York, NY, USA, 2004. ACM. ISBN 1-58113-951-9. doi: http://doi.acm.org/10.1145/1028509.1028520. URL http://doi.acm.org/10.1145/1028509.1028520.

Alex Rodriguez. Restful web services: The basics. IBM, Nov. 2008. URL https://www.ibm.com/developerworks/webservices/library/ws-restful/.

Huigui Rong, Ning Zhou, Hongqin Chen, and Hongli Cheng. Research on strategy of web service composition based on software life cycle. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, pages 1 –4, oct. 2008. doi: 10.1109/WiCom.2008.2002.

A.M. Ross, D.H. Rhodes, and D.E. Hastings. Using pareto trace to determine system passive value robustness. In *Systems Conference, 2009 3rd Annual IEEE*, pages 285 –290, march 2009. doi: 10.1109/SYSTEMS.2009.4815813.

G. Rothermel, R.H. Untch, Chengyun Chu, and M.J. Harrold. Test case prioritization: an empirical study. In *Software Maintenance, 1999. (ICSM '99) Proceedings. IEEE International Conference on*, pages 179 –188, 1999. doi: 10.1109/ICSM.1999.792604.

Ralph Schäfer. Rules for using multi-attribute utility theory for estimating a user's interests, workshop on adaptivity and user modelling. In *University of Dortmund*. Cambridge University Press, 2001.

S. Schmid, T. Chart, M. Sifalakis, and A. C. Scott. A highly flexible service composition framework for real-life networks. *Comput. Netw.*, 50:2488–2505, October 2006. ISSN 1389-1286. doi: 10.1016/j.comnet.2006.04.017. URL `http://portal.acm.org/citation.cfm?id=1163928.1163934`.

Hans Schuster, Dimitrios Georgakopoulos, Andrzej Cichocki, and Donald Baker. Modeling and composing service-based nd reference process-based multi-enterprise processes. In *Proceedings of the 12th International Conference on Advanced Information Systems Engineering*, CAiSE '00, pages 247–263, London, UK, UK, 2000. Springer-Verlag. ISBN 3-540-67630-9. URL `http://dl.acm.org/citation.cfm?id=646088.680058`.

A. Segev and E. Toch. Context-based matching and ranking of web services for composition. *Services Computing, IEEE Transactions on*, 2(3):210 –222, jul. 2009. ISSN 1939-1374. doi: 10.1109/TSC.2009.14.

Martin Shepperd and Gada Kadoda. Comparing software prediction techniques using simulation. *IEEE Trans. Softw. Eng.*, 27:1014–1022, November 2001. ISSN 0098-5589. doi: 10.1109/32.965341. URL `http://dl.acm.org/citation.cfm?id=506046.506050`.

Deng Shuiguang, Wu Zhaohui, and Li Ying. Ascend: A framework for automatic service composition and execution in dynamic environment. In *Proceedings of International Conference on Systems*, pages 3457–3461. IEEE Computer Society Press, 2004.

Forrest Shull, Guilherme Horta Travassos, Jeffrey Carver, and Victor Basili. Replicating software engineering experiments: Addressing the tacit knowledge problem, 2002.

Munindar P. Singh and Michael N. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. John Wiley and Sons, 2005.

Evren Sirin, Bijan Parsia, and James Hendler. Composition-driven filtering and selection of semantic web services. In *In AAAI Spring Symposium on Semantic Web Services*, page 2004, 2004.

Halvard Skogsrud, Boualem Benatallah, and Fabio Casati. Trust-serv: model-driven lifecycle management of trust negotiation policies for web services. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 53–62, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X. doi: http://doi.acm.org/10.1145/988672.988680. URL `http://doi.acm.org/10.1145/988672.988680`.

Haitao Song, Yingyu Yin, and Shixiong Zheng. Dynamic aspects weaving in service composition. *Intelligent Systems Design and Applications, International Conference on*, 1:1003–1008, 2006. doi: http://doi.ieeecomputersociety.org/10.1109/ISDA.2006.137.

Zoran Stojanovic and Ajantha Dahanayake. *Service-oriented Software System Engineering Challenges And Practices*. IGI Publishing, Hershey, PA, USA, 2005. ISBN 1591404274.

Agarwal Sudhir, Handschuh Siegfried, and Staab Steffen. Annotation, composition and invocation of semantic web services. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2(1):31–48, 2004.

Changlin Sun. Empirical reasoning about quality of service of component-based distributed systems. In *Proceedings of the 42nd annual Southeast regional conference*, ACM-SE 42, pages 341–346, New York, NY, USA, 2004. ACM. ISBN 1-58113-870-9. doi: http://doi.acm.org/10.1145/986537.986620. URL `http://doi.acm.org/10.1145/986537.986620`.

Loc Tran. Networking quality of service (QoS), October 2009. URL http://www.cbc.radio-canada.ca/technologyreview/pdf/issue6-QoS.pdf.

W. T. Tsai, Chun Fan, Yinong Chen, and Ray Paul. A service-oriented modeling and simulation framework for rapid development of distributed applications. *Simulation Modelling Practice and Theory*, 14(6):725–739, 2006.

W3C. Web services description language (wsdl) 1.1, March 2001. URL http://www.w3.org/TR/wsdl.

W3C. Web service choreography interface. W3C, August 2002. URL http://www.w3.org/TR/wsci/.

W3C. Web services choreography description language version 1.0. W3C, December 2004. URL http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/.

W3C. Soap version 1.2 part 0: Primer (second edition), April 2007. URL http://www.w3.org/TR/soap12-part0/.

Xiaoying Wang, Zhihui Du, and Yinong Chen. An adaptive model-free resource and power management approach for multi-tier cloud environments. *J. Syst. Softw.*, 85(5):1135–1146, May 2012. ISSN 0164-1212. doi: 10.1016/j.jss.2011.12.043. URL http://dx.doi.org/10.1016/j.jss.2011.12.043.

Yuanzhi Wang and K. Taylor. A model-driven approach to service composition. pages 8–13, dec. 2008. doi: 10.1109/SOSE.2008.42.

Paul Wernick and Tracy Hall. Getting the best out of software process simulation and empirical research in software engineering. In *Proceedings of the Second International Workshop on Realising Evidence-Based Software Engineering*, REBSE '07, pages 3–, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2962-3. doi: http://dx.doi.org/10.1109/REBSE.2007.3. URL http://dx.doi.org/10.1109/REBSE.2007.3.

R. J. Wieringa and J. M. G. Heerkens. The methodological soundness of requirements engineering papers: a conceptual framework and two

case studies. *Requir. Eng.*, 11(4):295–307, 2006. ISSN 0947-3602. doi: http://dx.doi.org/10.1007/s00766-006-0037-6.

Andrew B. Williams, Anand Padmanabhan, and M.B. Blake. Local consensus ontologies for B2B-oriented service composition. In *In AAMAS*, pages 647–654. Press, 2003.

Andrew B. Williams, Anand Padmanabhan, and M. Brian Blake. Experimentation with local consensus ontologies with implications for automated service composition. *IEEE Transactions on Knowledge and Data Engineering*, 17(7): 969–981, 2005.

Hang Wu and Chaozhen Guo. The research and implementation of web service classification and discovery based on semantic. In *Computer Supported Cooperative Work in Design (CSCWD), 2011 15th International Conference on*, pages 381 –385, june 2011. doi: 10.1109/CSCWD.2011.5960102.

Wang Xiaogang and Jia Tiejun. A web service composition model for multi-issues negotiation. In *Information Science and Engineering (ICISE), 2010 2nd International Conference on*, pages 949 –952, dec. 2010. doi: 10.1109/ICISE.2010.5688591.

Yu Xiaogao and Yu Xiaopeng. A knowledge-based approach for semantic service composition. In *Proceedings of Computational Engineering in Systems Applications CESA*, pages 1814–1821. IEEE Computer Society Press, 2006.

Chen Xinjun, Cai Wentong, S. J. Turner, and Wang Yong. Soar-dsgrid: Service-oriented architecture for distributed simulation on the grid. In *Proceeding of the 20th Principles of Advanced and Distributed Simulation workshop*, pages 65–73. IEEE Computer Society Press, 2006.

Jian Yang, Mike P. Papazoglou, and Willem-Jan Van den Heuvel. Tackling the challenges of service composition in e-marketplaces. In *Proceedings of the 12th International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems (RIDE'02)*, RIDE '02, pages 125–, Washington, DC, USA, 2002. IEEE Computer Society. URL `http://portal.acm.org/citation.cfm?id=882479.883718`.

Jian Yang, M.P. Papazoglou, B. Orriens, and W.-J. van Heuvel. A rule based approach to the service composition life-cycle. In *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*, pages 295 – 298, dec. 2003. doi: 10.1109/WISE.2003.1254497.

YanPing Yang, QingPing Tan, and Yong Xiao. Verifying web services composition based on hierarchical colored petri nets. In *Proceedings of the first international workshop on Interoperability of heterogeneous information systems*, IHIS '05, pages 47–54, New York, NY, USA, 2005. ACM. ISBN 1-59593-184-8. doi: http://doi.acm.org/10.1145/1096967.1096977. URL `http://doi.acm.org/10.1145/1096967.1096977`.

Chunyang Ye, S.C. Cheung, and W.K. Chan. Publishing and composition of atomicity-equivalent services for B2B collaboration. In *In ICSE 06*, pages 351–360. ACM, 2006.

Robert Yin. *Case Study Research: Design & Methods*. Sage Books, 3 edition, 2003.

Fei You, Qingxi Hu, Yuan Yao, Gaochun Xu, and Minglun Fang. Study on web service matching and composition based on ontology. In *Computer Science and Information Engineering, 2009 WRI World Congress on*, volume 4, pages 542 –546, 31 2009-april 2 2009. doi: 10.1109/CSIE.2009.673.

Muhammad Younas, Kuo-Ming Chao, and Christopher Laing. Composition of mismatched web services in distributed service oriented design activities. *Adv. Eng. Inform.*, 19:143–153, April 2005. ISSN 1474-0346. doi: 10.1016/j.aei.2005.05.009. URL `http://portal.acm.org/citation.cfm?id=1640987.1641047`.

Muhammad Younas, Irfan Awan, and David Duce. An efficient composition of web services with active network support. *Expert Systems with Applications*, 31(4):859 – 869, 2006. ISSN 0957-4174. doi: DOI: 10.1016/j.eswa.2006.01.008. URL `http://www.sciencedirect.com/science/article/B6V03-4J3WNVF-5/2/9eeef167b09- 599faf4be3127f534c6d9`. Computer Supported Cooperative Work in Design and Manufacturing.

Tao Yu, Yue Zhang, and Kwei-Jay Lin. Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Trans. Web*, 1 (1), May 2007. ISSN 1559-1131. doi: 10.1145/1232722.1232728. URL `http://doi.acm.org/10.1145/1232722.1232728`.

Maamar Zakaria, Sheng Quan Z., Benatallah Boualem, and Al-Khatib Ghazi. A three-level specification approach for an environment of software agents and web services. *Electronic Commerce Research and Applications*, 3(3):214–231, 2004.

L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng. Quality driven web services composition. In *Proceedings of the 12th international Conference on World Wide Web*, pages 411–421. ACM Press, 2003.

Yafeng Zheng, Songyang Ding, and Xiaoxiao Zhang. Key technology research on dynamic web service composition based on semantic web. pages 1068 –1071, Dec. 2008. doi: 10.1109/KAMW.2008.4810677.

Xiaomin Zhu, Chuan He, Rong Ge, and Peizhong Lu. Boosting adaptivity of fault-tolerant scheduling for real-time tasks with service requirements on clusters. *J. Syst. Softw.*, 84(10):1708–1716, October 2011. ISSN 0164-1212. doi: 10.1016/j.jss.2011.04.067. URL `http://dx.doi.org/10.1016/j.jss.2011.04.067`.

Appendix A

The outcomes from SLR

A.1 Research Method

The description of research methods used in this papers are summarised in Table A.1.

A.2 Tables of practical solutions for DWSC but with no formal evaluations

In this appendix we summaries those papers that described practical solutions for DWSC, but which lacked any formal evaluation (concept implementation). The different categories are described in Tables A.2to A.7.

Table A.1: Categories of Research Method defined in Brereton et al. (2006)

RESEARCH METHOD	**DESCRIPTION**
AR - Action Research	Researcher participates in the action studied. Differs from participant observation as researcher is aware that their presence will affect the situation they are researching
CA - Conceptual Analysis	Basic assumptions behind constructs are first analysed; theories, models and frameworks used in previous empirical studies are identified and logical reasoning is thereafter applied
CAM - Conceptual Analysis/mathematical	CA using mathematical techniques.
CI - Concept Implementation / proof of concept.	Self-explanatory - 'We built it and it worked'
CS - Case Study	A method, tool or procedure under investigation is tried out on a real project using the (otherwise) standard methods/tools/procedures of the organisation
DA - Data Analysis	Analysis of data generated or published elsewhere
ET - Ethnography	Researcher participates in the action studied. Differs from AR in that researcher has no intent of interfering in phenomenon, does not relate findings to generalisable theory and does not interpret from researcher's point of view
FE - Field Experiment	Extensions of laboratory experiments into the 'real world' of organisations/society. Independent variables are controlled
FS - Field Study	Examines data collected from several projects (or subjects) simultaneously. Less intrusive than case study. Usually less detailed than case study because aim is not to perturb the subject under study
ID - instrument Development	Development of MIS (or other) instrument (e.g. user satisfaction questionnaire)
LH - Laboratory Experiment - Human Subjects	Identification of precise relationships between variables in a designed controlled environment (i.e. a laboratory) using human subjects and quantitative techniques
LR - Literature Review/analysis	Examine/analyse previous publication [1]
LS - Laboratory Experiment - Software	A laboratory experiment to compare the performance of newly proposed system with other (existing) systems
MP - Mathematical Proof	Self-explanatory
PA - Protocol Analysis	Used to reduce the large amount of data generated by use of 'think aloud'. Requires the production of a protocol (a categorisation of the possible relevant utterances) and the application of that protocol to gathered data
SI - Simulation	Execution of a system with artificial data, using a model of the real world [3]
ES - Descriptive/Exploratory survey	An exploratory field study in which there is no test of relationships between variables

Table A.2: Realisation of Semantic Web and Ontology strategies for Service Composition.

Author(s)	Interpretation of Strategy	Form of Realisation	Observations Provided
Hamadi and Benatallah (2003)	Used formal semantics with Petri net-based algebra for composing Web services.	Framework	None
Carnegie and Paolucci (2003)	Proposed a Web Service architecture that takes advantage of DAML-S information to support automatic discovery and interaction between Web Services.	Implementation	No details about its use
Dong et al. (2006)	Examined the association between BPEL and HPNs for modelling the use request, which also helps with analysing the composition behaviour.	MIS tool	Model the Workflow
Dumez et al. (2008)	Presented a UML-S (UML for Services) which is a new UML-based formalism to develop composite Web services according to MDA principles.	Case study	None
Fujii and Suda (2004b)	Presented a semantics-based dynamic service composition system that consists of CoSMoS (Component Service Model with Semantics), CoRE (Component Runtime Environment) and SeGSeC (Semantic Graph based Service Composition).	Implemented a MIS tool	Conceptual analysis.
Chunming et al. (2006)	Focused on verification of the produced model through checking of its properties and control flow.	MIS tool	Extended pi-calculus
Jianhong et al. (2004)	A semantic type-matching algorithm is introduced to match actions of the plan to suitable operations of Web services.	Example as use case	None
Kang and Sohn (2006)	Presented a service template authoring environment for the URC service composition system in the semantic Web services environments.	Implementation	None
In-Young et al. (2002)	Described the DWSC tool that employs a dynamic coordination mechanism to facilitate the creation of information management applications for processing dynamic Web content.	Implemented MIS tool	Example
Lu et al. (2006)	Extended OWL-S by introducing the service assumption to provide a more comprehensive service description schema.	An illustrated example	Extended form of OWL-S
Narayanan and McIlraith (2002)	Provided a model-theoretic semantics as well as a distributed operational semantics that can be used for simulation, validation, verification, automated composition and enactment of DAML-S-described Web services.	Implementation	Partial simulation
Narayanan and McIlraith (2003)	This paper is an application of their previous work which uses model-theoretic semantics as well as a distributed operational semantics.	Implemented MIS tool	Conceptual analysis.
Nie et al. (2006)	Presented an approach for composing Web services on demand. Used Web semantics to match service functions and the SLA description is applied to QoS negotiation.	An illustrated example	Description language
Rao et al. (2006)	Described an approach to automatic Semantic Web service composition. The approach has been directed to provided for automated composition and semantic composition. Their method was used Linear Logic (LL) theorem proving.	Prototype	Semantic composition
Sirin et al. (2004)	Used semantic descriptions through an ontology to aid in the composition of Web services. They presented a goal-oriented, interactive composition approach utilising matchmaking algorithms.	Prototype	OWL-S
Skogsrud et al. (2004)	Described Trust-Serv, a trust negotiation framework for access control in Web services.	Implementation	No details about any experiments
Ye et al. (2006)	Presented a novel approach to publishing and discovering services with atomicity sphere consistency for B2B collaboration.	Scenario as use case	None

Table A.3: Realisation of QoS for Service Composition strategies.

Author(s)	Interpretation of Strategy	Form of Realisation	Observations Provided
Chaari et al. (2008)	Presented QoS-based policies for Web service selection. It extends the WS-policy standard with some ontological concepts to represent QoS attributes.	Conceptual framework	Ontology use
Di Penta et al. (2006)	Developed Web service Binder, which is a framework that supports the dynamic binding of composite services. Bindings are determined according to functional policies and to global and local QoS optimization criteria	Non-linear QoS aggregation formulas	Faults recovery

Table A.4: Realisation of Using Knowledge-based forms for service composition.

Author(s)	Interpretation of Strategy	Form of Realisation	Observations Provided
Ito and Tanaka (2003)	Proposed a new framework for visually and dynamically defining functional linkages among Web applications to compose a single application tool.	Implementation	WS wrapper
Qiu et al. (2006)	Proposed an algorithm for Service composition using a backward-chaining search method to find potential candidate services.	Prototype	Use case as example
Xiaogao and Xiaopeng (2006)	Discussed a novel, knowledge-based approach to resource synthesis and provided advice on service selection and instantiation.	Prototype	Semantic/Workflow

Table A.5: Realisation of Workflow and Languages for service composition.

Author(s)	Interpretation of Strategy	Form of Realisation	Observations Provided
Baresi et al. (2004)	Presented an approach for monitoring the execution of composed Web services. The approach starts from BPEL processes: The user annotates the process and the system transforms it into a monitored process.	Use case	Language
Casati et al. (2000)	The HP Laboratory has proposed eFlow as framework to support composite e-services specification and management, which are modelled a processes that will performed by a service process engine.	Platform	Workflow
Geebelen et al. (2008)	Presented a framework that allows the design of WS-BPEL processes in a modular way based on reusable templates. In addition, introduced an extra layer on top of WS-BPEL that allows template processing based on parameter values.	Prototype	Framework
Kapitsaki et al. (2008)	Presented an approach for the modelling of the application using UML class and state transition diagrams and the transformation to appropriate platform specific code.	MIS tool	Workflow
Leymann (2008)	Introduced Web services flow language (WSFL 1.0) features which can be used to create new Web services from existing Web services.	Illustrated examples	Provided a new language
Liu et al. (2005)	Developed a new domain-specific visual language called ViTABaL-WS to support modelling of complex interactions between Web service components.	Prototype	MIS tool
Liu et al. (2004b)	Proposed an OPD-based approach for Web Service composition; it introduced several mapping rules between OPD and BPEL4WS, and also extends OPD with the equivalent link and proposes a set of OPD templates to represent Web Service composition originally described using BPEL4WS.	MIS tool	Modelling language
Lopes et al. (2008)	Devised a framework which is a Web-based application which supports basic ideas and rules of Web2.0 technology. These principles support the coordination and integration of the service and data sources; and simplify the extraction of knowledge in various scientific fields.	Conceptual framework	Workflow
Yang et al. (2005)	Introduces an approach to verify and analyse Web Services composition based on transformation composition specification to CP-nets (Coloured Petri Nets).	Prototype	Study
Yang et al. (2002)	Presented a framework to discuss the different forms of service composition and their essential characteristics.	MIS tool	Specification language (SCSL)

Table A.6: Realisation of Middleware for Service Composition.

Author(s)	Interpretation of Strategy	Form of Realisation	Observations Provided
Kalasapur et al. (2006)	Presented a mechanism to evaluate service oriented architectures for pervasive computing systems.	Middleware	Partial simulation
Limam et al. (2007)	Presented *OSDA*, a novel cross-domain service discovery architecture. It allows the service providers and consumers seamless access to service and resource discovery across domains using local discovery mechanisms.	Prototype	Case scenario
Jin and Hideyuki (2002)	Developed *Serdget Markup Language* (SML) to support developers with building heterogeneous services on the Virtual Networked Appliance (VNA) architecture. (serdget = service gadget which is a component in their architecture)	Conceptual framework	Middleware architecture
Robinson et al. (2004)	Introduced *Scooby* middleware for WSC in pervasive computing, it offers a composition language for the service management and organisation.	Initial design and implementation	Middleware language

Table A.7: Realisation of Agent software for Service Composition.

Author(s)	Interpretation of Strategy	Form of Realisation	Observations Provided
Abrougui et al. (2009)	Presented a *Recursive multi-agent system* (RMAS), a model for dynamic and adaptive WSC. The model is composed of application, business and composition layers, each of those layers has been represented by an agent and each agent may be seen as a multi-agent system.	Prototype	None
Cao et al. (2004)	Proposed a service-based reconfigurable system which is based on Workflow and service agents.	Prototype	Case study
Kosuga et al. (2002)	Presented a multimedia service composition scheme for sharing computer and communication resources by using plural terminals in the ubiquitous computing environment.	Prototype	Agent+QoS

Appendix B

Results from the Close Replication Study Experiments

B.1 First Experiment: QoS Metrics of the Selected Plans

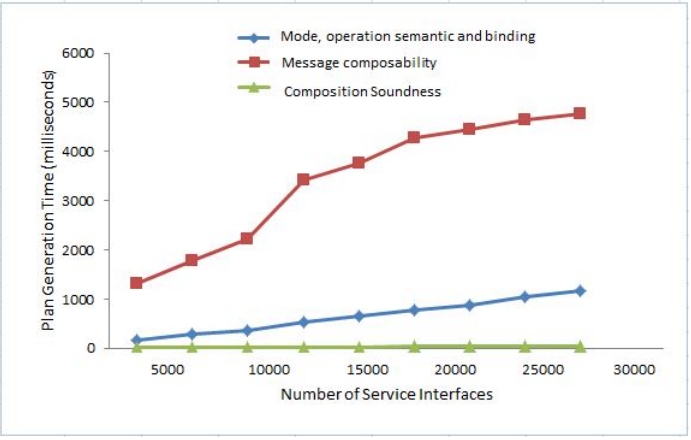


Figure B.1: Plan generation time: Replication study - Result 1

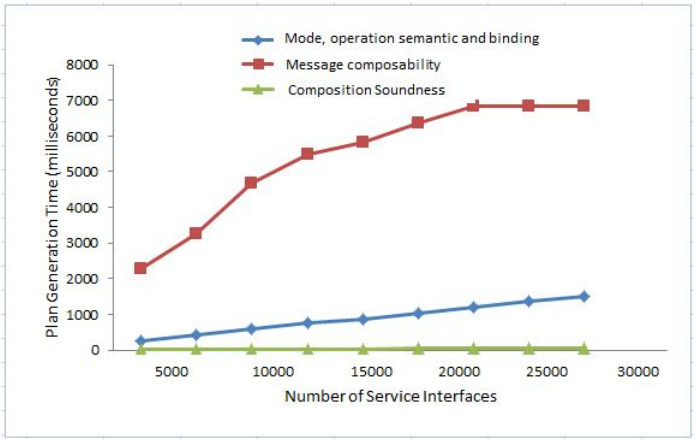


Figure B.2: Plan generation time: Replication study - Result 2

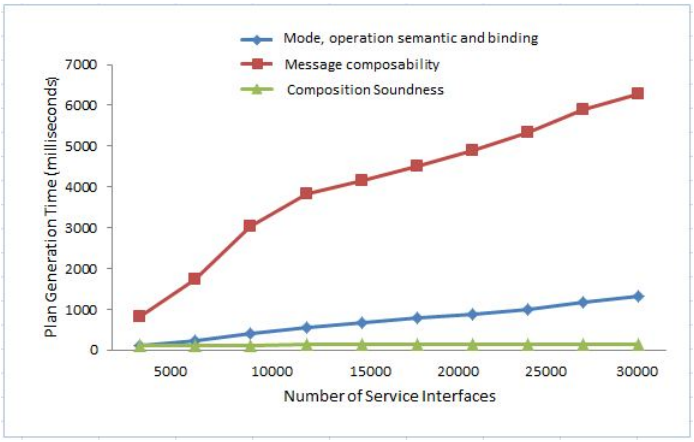


Figure B.3: Plan generation time: Replication study - Result 3

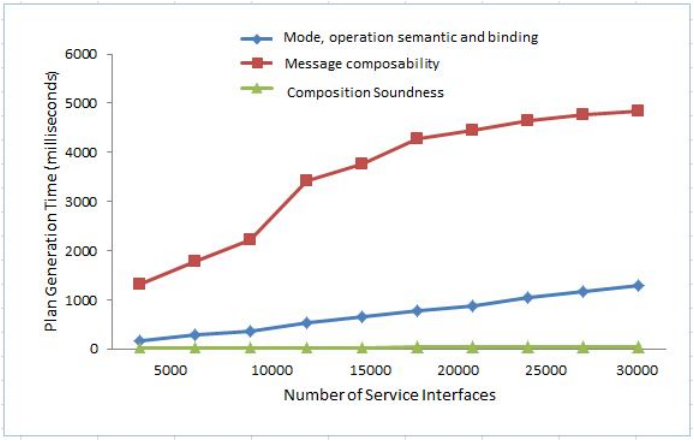


Figure B.4: Plan generation time: Replication study - Result 4

B.2 Second Experiment: Testing the Composite Completeness (CC) Ratio

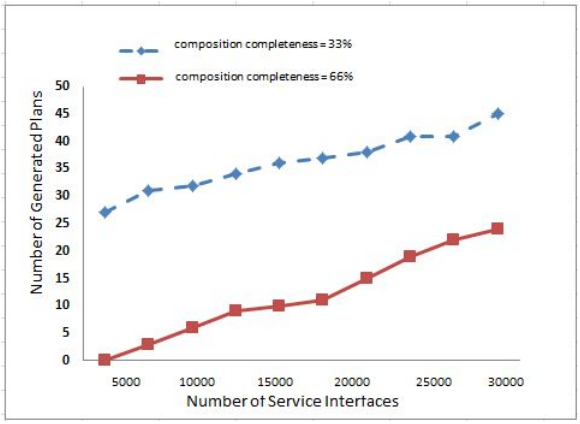


Figure B.5: Number of generated plans - Result 1

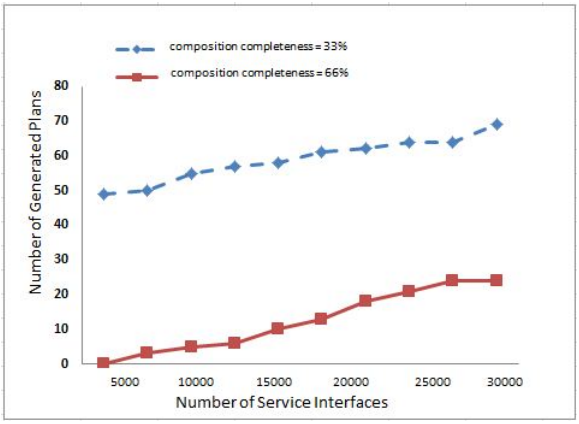


Figure B.6: Number of generated plans - Result 2

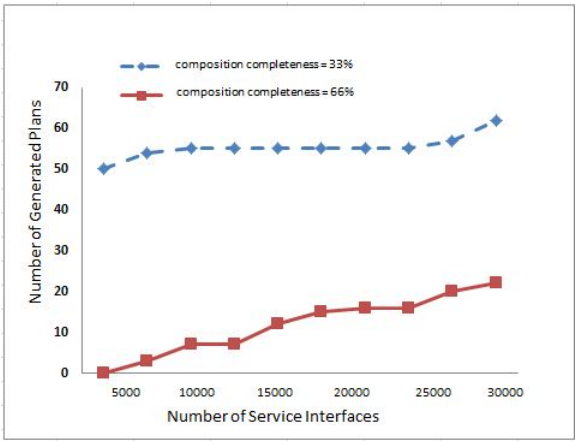


Figure B.7: Number of generated plans - Result 3

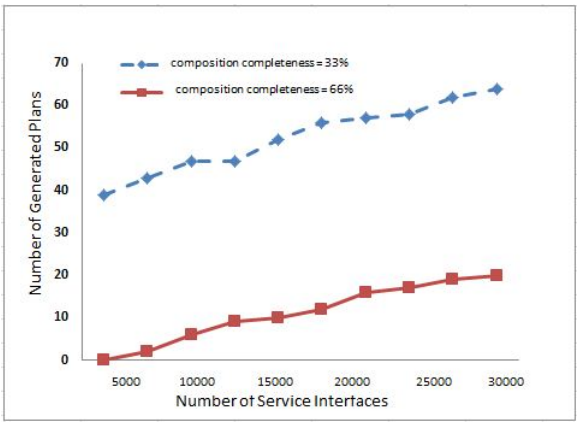


Figure B.8: Number of generated plans - Result 4

Appendix C

Results from Service Selection Experiments

C.1 First Experiment: QoS Metrics of the Selected Plans

	QoS model					Overall Score	
Instance No	Execution Price	Execution duration	Reputation	Reliability	Availability	QoS	Ranking
1	2.422	40.808	5.049	1.001	0.924	0.782	0.401
2	1.979	16.779	5.182	0.64	0.901	0.711	0.452
3	6.03	23.317	4.742	0.979	0.888	0.752	0.21
4	1.665	23.515	4.95	0.913	0.957	0.864	0.448
5	5.661	13.285	4.857	0.91	0.858	0.762	0.594
6	1.049	27.584	5.06	0.95	0.903	0.827	0.348
7	2.397	28.81	5.857	0.747	1.002	0.774	0.343
8	1.033	11.538	3.567	0.797	0.975	0.821	0.344
9	2.75	25.492	5.955	0.986	0.987	0.878	0.878
10	1.386	10.279	5.881	0.892	0.926	0.914	0.608
Average	**2.6372**	**22.1407**	**5.11**	**0.8815**	**0.9321**	**0.8085**	**0.4626**

Table C.1: Comparison between Ranking and QoS model techniques - Result 1

	QoS model					Overall Score	
Instance No	Execution Price	Execution duration	Reputation	Reliability	Availability	QoS	Ranking
1	10.619	20.216	1.829	0.746	0.92	0.754	0.56
2	10.403	49.716	1.793	0.517	0.979	0.807	0.252
3	10.125	17.821	5.069	0.734	0.546	0.817	0.524
4	10.728	33.575	2.711	0.762	0.897	0.826	0.454
5	10.998	30.3	1.617	0.542	0.707	0.819	0.495
6	10.765	41.878	2.652	0.751	0.72	0.847	0.502
7	10.568	23.88	3.444	0.698	0.806	0.788	0.602
8	10.98	40.257	2.534	0.879	0.784	0.853	0.751
9	10.904	13.774	3.894	0.89	0.828	0.812	0.302
10	10.978	19.298	4.624	0.67	0.759	0.775	0.723
Average	**10.7068**	**29.0715**	**3.0167**	**0.7189**	**0.7946**	**0.8098**	**0.5165**

Table C.2: Comparison between Ranking and QoS model techniques - Result 2

	QoS model					Overall Score	
Instance No	Execution Price	Execution duration	Reputation	Reliability	Availability	QoS	Ranking
1	2.422	40.808	5.049	1.001	0.924	0.401	0.782
2	1.979	16.779	5.182	0.64	0.901	0.452	0.711
3	6.03	23.317	4.742	0.979	0.888	0.21	0.752
4	1.665	23.515	4.95	0.913	0.957	0.448	0.864
5	5.661	13.285	4.857	0.91	0.858	0.594	0.762
6	1.049	27.584	5.06	0.95	0.903	0.348	0.827
7	2.397	28.81	5.857	0.747	1.002	0.343	0.774
8	1.033	11.538	3.567	0.797	0.975	0.344	0.821
9	2.75	25.492	5.955	0.986	0.987	0.878	0.878
10	1.386	10.279	5.881	0.892	0.926	0.608	0.914
Average	**26.372**	**221.407**	**51.1**	**8.815**	**9.321**	**4.626**	**8.085**

Table C.3: Comparison between Ranking and QoS model techniques - Result 3

	QoS model					Overall Score	
Instance No	Execution Price	Execution duration	Reputation	Reliability	Availability	QoS	Ranking
1	10.211	39.598	4.174	0.617	0.628	0.824	0.824
2	10.929	30.669	0.117	0.647	0.787	0.79	0.323
3	10.526	11.878	0.165	0.622	0.875	0.783	0.759
4	10.854	50.767	5.291	0.805	0.59	0.782	0.38
5	10.742	42.126	5.272	0.672	0.624	0.752	0.346
6	10.953	48.315	3.904	0.776	0.93	0.806	0.47
7	10.796	29.01	3.857	0.746	0.685	0.809	0.335
8	10.965	46.913	0.562	0.662	0.648	0.78	0.632
9	10.99	13.595	1.117	0.924	0.911	0.737	0.511
10	10.963	31.746	3.74	0.665	0.962	0.835	0.521
Average	**10.7929**	**34.4617**	**2.8199**	**0.7136**	**0.764**	**0.7898**	**0.5101**

Table C.4: Comparison between Ranking and QoS model techniques - Result 4

C.2 Second Experiment: Computation Time

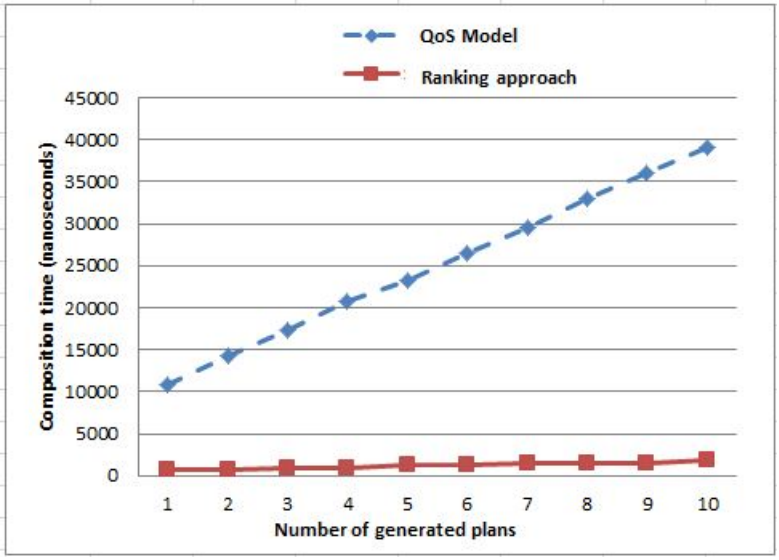


Figure C.1: Comparison between Ranking and QoS model techniques - Result 1

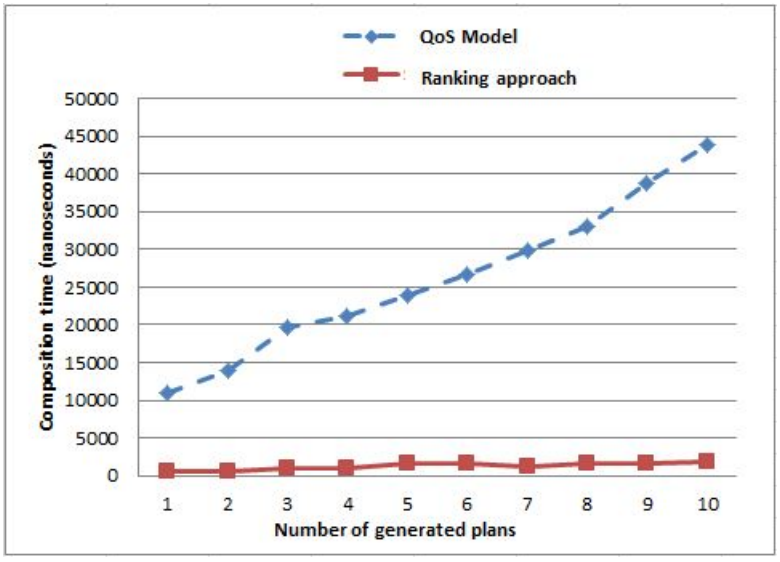


Figure C.2: Comparison between Ranking and QoS model techniques - Result 2

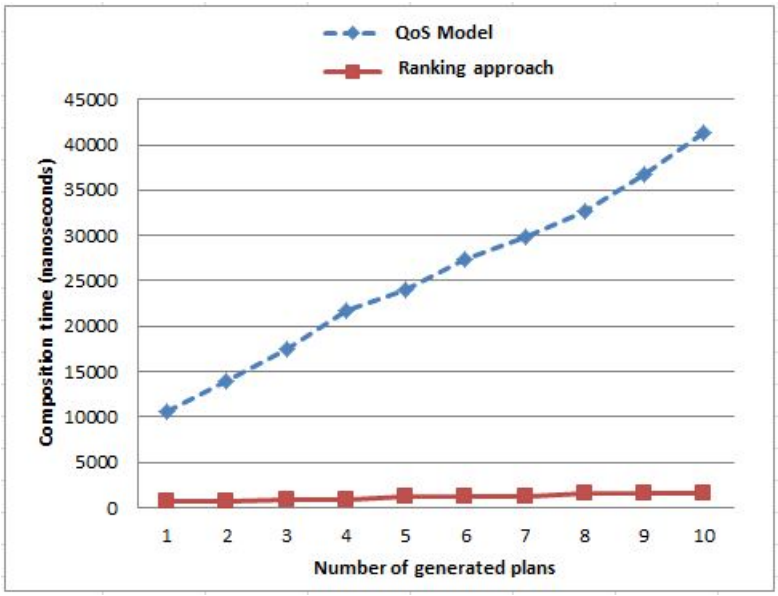


Figure C.3: Comparison between Ranking and QoS model techniques - Result 3

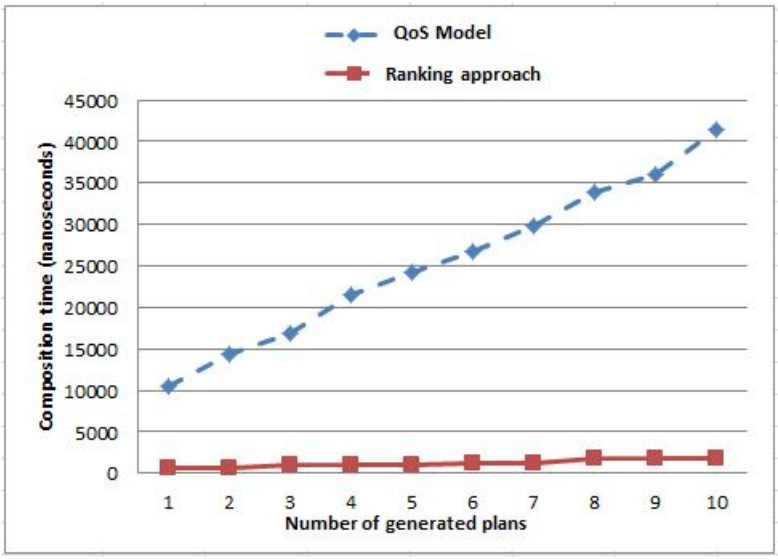


Figure C.4: Comparison between Ranking and QoS model techniques - Result 4